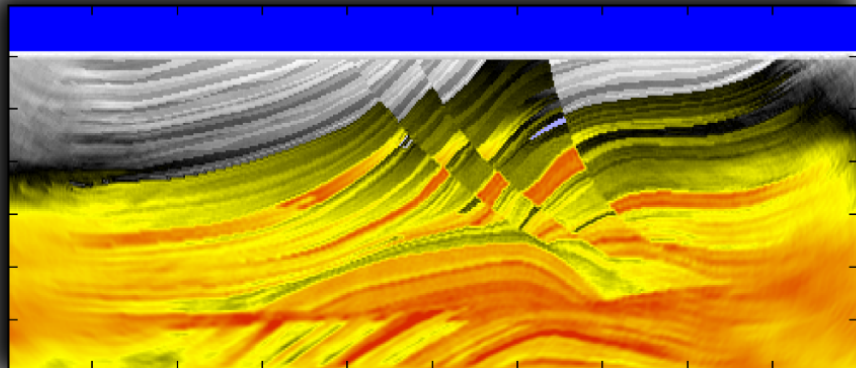
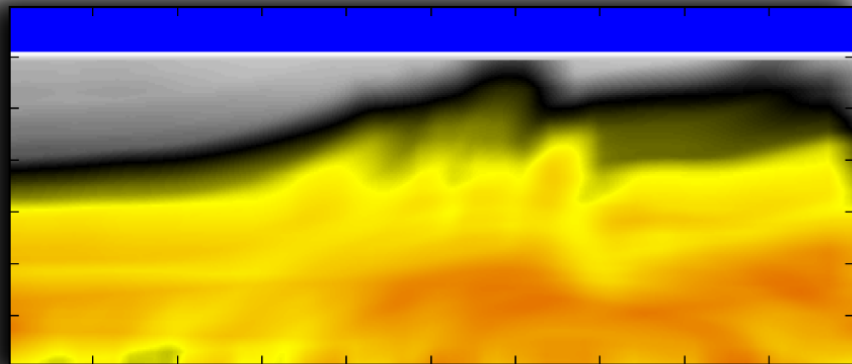




DENISE

User Manual



Daniel Köhn
Denise De Nil
André Kurzmann
Thomas Bohlen
Lisa Groos
Martin Schäfer
Sven Heider
Linbin Zhang



DENISE

User Manual

© Christian-Albrechts-Universität Kiel (Germany) and
Karlsruher Institut für Technologie (KIT) (Germany)
Version 1.1

April 24, 2015

Authors

The DENISE code was first developed by Daniel Köhn, Denise De Nil and André Kurzmann at the Christian-Albrechts-Universität Kiel and TU Bergakademie Freiberg (Germany) from 2005 to 2009.

The forward code is based on the viscoelastic FD code fdveps (now SOFI2D) by Bohlen [2002].

Different external libraries for timedomain filtering are used.

The copyright of the source codes are held by different persons:

cseife.c, cseife.h, lib_stfinv, lib_aff, lib_fourier:

Copyright (c) 2005 by Thomas Forbriger (BFO Schiltach)

cseife_deriv.c, cseife_gauss.c, cseife_rekfl.c, cseife_rfk.c and cseife_tides.c:

Copyright (c) 1984 by Erhard Wielandt

This algorithm was part of seife.f. A current version of seife.f can be obtained from <http://www.software-for-seismometry.de/>

The Matlab implementation of a few SU routines, mainly used to read and write SU files in data pre-processing are:

Copyright (C) 2008, Signal Analysis and Imaging Group

For more information: <http://www-geo.phys.ualberta.ca/saig/SeismicLab>

Author: M.D.Sacchi

Since then it has been further developed, maintained and improved by contributions from: in alphabetical order,

Lisa Groos,

Sven Heider,

Martin Schäfer,

Linbin Zhang ...

(add other developers here in the future).

License

DENISE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 2.0 of the License only. DENISE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with DENISE. See file COPYING and/or <http://www.gnu.org/licenses/gpl-2.0.html>. The authors of DENISE are listed in the AUTHORS section.

Contents

1	Introduction	4
1.1	Citation	5
1.2	Support	5
2	Theoretical Background	7
2.1	Equations of motion for an elastic medium	7
2.2	Solution of the elastic wave equation by finite differences	8
2.2.1	Discretization of the wave equation	8
2.2.2	Accuracy of FD operators	9
2.2.3	Initial and Boundary Conditions	10
2.3	Numerical Artefacts and Instabilities	13
2.3.1	Grid Dispersion	13
2.3.2	The Courant Instability	16
3	The adjoint problem	18
3.1	What is an "optimum" model ?	18
3.2	How to find an optimum model	19
3.3	Calculation of the gradient direction $\frac{\partial E}{\partial \mathbf{m}}$	20
3.4	Gradients for different model parametrizations	26
3.5	Gradients for the stress-velocity code	26
3.6	Estimation of an optimum step length μ_n	27
3.7	Nonlinear Conjugate Gradient Method	29
3.8	The elastic FWT algorithm	32
4	Source Wavelet Inversion	33
5	Getting Started	35
5.1	Requirements	35
5.1.1	LAM	35
5.1.2	How to run DENISE on the NEC-Linuxcluster at RZ Kiel	36
5.2	Installation	37
5.3	Compilation of DENISE	38
5.4	Running the program	40
5.5	Postprocessing	41
6	Definition of parameters for the modeling and inversion code	43
6.1	Input file with fixed parameters DENISE.inp	43
6.2	Workflow file with variable inversion parameters FWI_workflow.inp	58

7	Example 1 - the elastic Marmousi2 model	62
7.1	The complex Marmousi2 model	62
7.1.1	Acquisition geometry and FD model	65
7.1.2	Elastic wave propagation in the complex Marmousi model	65
7.1.3	FWI of the complex Marmousi model	68
7.1.4	Marmousi-II Benchmarks	75

Chapter 1

Introduction

The aim of Full Waveform Tomography (FWT) is to estimate the elastic material parameters in the underground. This can be achieved by minimizing the misfit energy between the modelled and field data using a gradient optimization approach. Because the FWT uses the full information content of each seismogram, structures below the seismic wavelength can be resolved. This is a tremendous improvement in resolution compared to traveltome tomography (Pratt et al. [2002]).

The concept of full waveform tomography was originally developed by Albert Tarantola in the 1980s for the acoustic, isotropic elastic, and viscoelastic case (Tarantola [1984b,a, 1986, 1988]). First numerical implementations were realized at the end of the 1980s (Gauthier et al. [1986], Mora [1987], Pica et al. [1990]), but due to limited computational resources, the application was restricted to simple 2D synthetic test problems and small near offset datasets. At the beginning of the 1990s the original time domain formulation was transferred to a robust frequency domain approach (Pratt and Worthington [1990], Pratt [1990]). With the increasing performance of supercomputers moderately sized problems could be inverted with frequency domain approaches.

A spectacular result to prove the application of acoustic FWT on laboratory scale was presented by Pratt [1999] for ultrasonic tomography measurements on a simple block model. In a numerical blind test Brenders and Pratt [2007] achieved a very good agreement between their inversion result and the unknown true P-wave velocity model. The parallelization and performance optimizations of the frequency domain approach (see e.g. Sourbier et al. [2009a], Sourbier et al. [2009b]) lead to a wide range of acoustic FWT applications for problems on different scales, from the global scale, crustal scale over engineering and near surface scale, down to laboratory scale (Pratt [2004]).

Beside the application to geophysical problems, the acoustic FWT is also used to improve the resolution in medical cancer diagnostics (Pratt et al. [2007]). However, all these examples are restricted to the inversion of the acoustic material parameters: P-wave velocity, density and additionally the viscoacoustic damping Q_p for the P-waves. Even today the independent 2D FWT of all three isotropic elastic material parameters is still a challenge. Most elastic approaches invert for P-wave velocity only and use empirical relationships to deduce the distribution of S-wave velocity and density (Shipp and Singh [2002], Sheen et al. [2006]). Recently some authors also investigated the independent multiparameter FWT in the frequency domain (Choi et al. [2008a,b], Brossier [2009]).

In order to extract information about the structure and composition of the crust from seismic observations, it is necessary to be able to predict how seismic wavefields are affected by complex structures. Since exact analytical solutions to the wave equations do not exist for most subsurface configurations, the solutions can be obtained only by numerical methods. For iterative calculations of synthetic seismograms with limited computer resources fast and accurate modeling methods are needed.

The FD modeling/inversion program DENISE (subwavelength **DE**tail resolving **Nonlinear Iterative SE**ismic inversion), is based on the FD approach described by Virieux [1986] and Levander [1988]. The present program DENISE has the following extensions

- is efficiently parallelized using domain decomposition with MPI Bohlen [2002],
- considers viscoelastic wave propagation effects like attenuation and dispersion Robertsson et al. [1994], Blanch et al. [1995], Bohlen [2002],
- employs higher order FD operators,

- applies Convolutional Perfectly Matched Layer boundary conditions at the edges of the numerical mesh Komatitsch and Martin [2007].

In the following sections, we give an extensive description of the theoretical background, the different input parameters and show a few benchmark modeling and inversion applications.

1.1 Citation

If you use this code for your own research, please cite at least one article written by the developers of the package, for instance:

Köhn [2011]

Köhn et al. [2012]

or

(XX add more references here)

and/or other articles from (<http://www.geophysik.uni-kiel.de/~dkoehn/publications.htm>)

The corresponding BibTeX entries may be found in file `doc/USER_MANUAL/thesis.bib`.

1.2 Support

The development of the code was supported by the Christian-Albrechts-Universität Kiel, TU Bergakademie Freiberg, Deutsche Forschungsgemeinschaft (DFG), Bundesministerium für Bildung und Forschung (BMBF), Bundesministerium für Umwelt, Naturschutz, Bau und Reaktorsicherheit (BMU), the Wave Inversion Technology (WIT) Consortium and the Verbundnetz-Gas AG (VNG). The code was tested and optimized at the computing centres of Kiel University, TU Bergakademie Freiberg, TU Chemnitz, TU Dresden, the Karlsruhe Institute of Technology (KIT) and the Hochleistungsrechenzentrum Nord (HLRN 1+2).

Acknowledgments and contact

We thank for constructive discussions and further code improvements:

Anna Przebindowska (Karlsruhe Institute of Technology),
Olaf Hellwig (TU Bergakademie Freiberg),
Dennis Wilken and Wolfgang Rabbel (Christian-Albrechts-Universität Kiel).

Please e-mail your feedback, questions, comments, and suggestions to
Daniel Köhn (`dkoehn-AT-geophysik.uni-kiel.de`).

Chapter 2

Theoretical Background

2.1 Equations of motion for an elastic medium

The propagation of waves in a general elastic medium can be described by a system of coupled linear partial differential equations. They consist of the equations of motion

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i \quad (2.1)$$

which simply state that the momentum of the medium, the product of density ρ and the displacement velocity v_i , can be changed by surface forces, described by the stress tensor σ_{ij} or body forces f_i . These equations describe a general medium, like gas, fluid, solid or plasma. The material specific properties are introduced by additional equations which describe how the medium reacts when a certain force is applied. In the isotropic elastic case this can be described by a linear stress-strain relationship:

$$\begin{aligned} \sigma_{ij} &= \lambda \theta \delta_{ij} + 2\mu \epsilon_{ij} \\ \epsilon_{ij} &= \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \end{aligned} \quad (2.2)$$

where λ and μ are the Lamé parameters, ϵ_{ij} the strain tensor and u_i the displacement. Using $v_i = \frac{\partial u_i}{\partial t}$, (2.1) and (2.2) can be transformed into a system of second order partial differential equations:

$$\begin{aligned} \rho \frac{\partial^2 u_i}{\partial t^2} &= \frac{\partial \sigma_{ij}}{\partial x_j} + f_i \\ \sigma_{ij} &= \lambda \theta \delta_{ij} + 2\mu \epsilon_{ij} \\ \epsilon_{ij} &= \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \end{aligned} \quad (2.3)$$

This expression is called **Stress-Displacement** formulation. Another common form of the elastic equations of motion can be deduced by taking the time derivative of the stress-strain relationship and the strain tensor in Eq. (2.3). Since the Lamé parameters λ and μ do not depend on time, Eq. (2.3) can be written as:

$$\begin{aligned} \rho \frac{\partial v_i}{\partial t} &= \frac{\partial \sigma_{ij}}{\partial x_j} + f_i \\ \frac{\partial \sigma_{ij}}{\partial t} &= \lambda \frac{\partial \theta}{\partial t} \delta_{ij} + 2\mu \frac{\partial \epsilon_{ij}}{\partial t} \\ \frac{\partial \epsilon_{ij}}{\partial t} &= \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \end{aligned} \quad (2.4)$$

This expression is called **Stress-Velocity** formulation. For simple cases (2.3) and (2.4) can be solved analytically. More complex problems require numerical solutions. One possible approach for a numerical solution is described in the next section.

2.2 Solution of the elastic wave equation by finite differences

2.2.1 Discretization of the wave equation

For the numerical solution of the elastic equations of motion, Eqs. (2.3) have to be discretized in time and space on a grid. The particle displacement $\mathbf{u}$, the stresses σ_{ij} , the Lamé parameters λ and μ are calculated and defined at discrete Cartesian coordinates $x = i \, dh$, $y = j \, dh$ and discrete times $t = n \, dt$. dh denotes the spatial distance between two adjacent grid points and dt the difference between two successive time steps. Therefore every grid point is located in the interval $i \in \mathbb{N}[1, NX]$, $j \in \mathbb{N}[1, NY]$ and $n \in \mathbb{N}[1, NT]$, where NX , NY and NT are the number of discrete spatial grid points and time steps, respectively. Finally the partial derivatives are replaced by **finite-difference (FD)** operators. Two types of operators can be distinguished, forward and backward operators D^+ , D^- . The derivative of a function y with respect to a variable x can be approximated by the following operators:

$$\begin{aligned} D_x^+ y &= \frac{y[i+1] - y[i]}{dh} && \text{forward operator} \\ D_x^- y &= \frac{y[i] - y[i-1]}{dh} && \text{backward operator} \end{aligned} \quad (2.5)$$

To calculate the spatial derivatives of the wavefield variables at the correct positions, the variables are not placed on the same grid points, but staggered by half of the spatial grid point distance (Virieux [1986] and Levander [1988]). figure 2.1 shows the distribution of the material parameters and wavefield variables on the spatial grid.

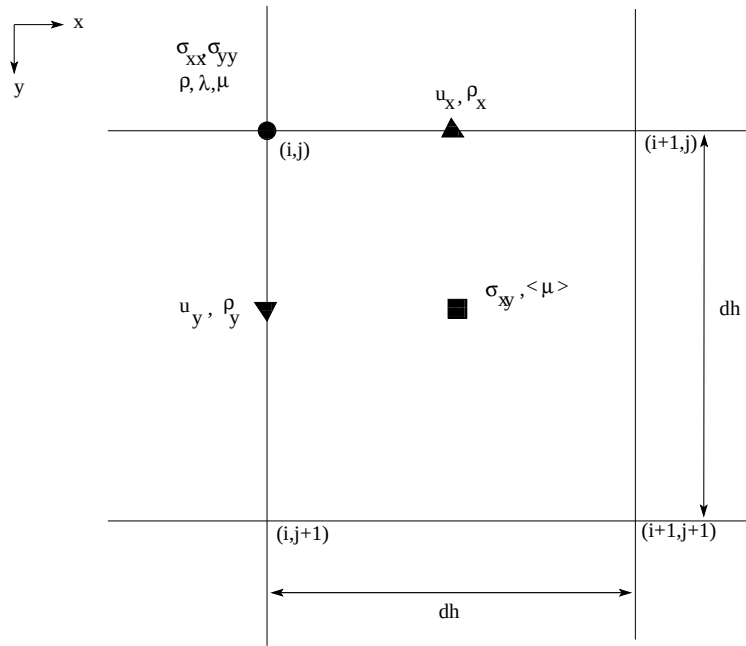


Figure 2.1: Grid geometry for a standard staggered grid (SSG) in Cartesian coordinates as suggested by Virieux [1986] and Levander [1988].

To guarantee the stability of the **standard staggered grid (SSG)** code, the Lamé parameter μ and density ρ have to be averaged harmonically and arithmetically (Moczso et al. [2004], Bohlen and Saenger [2006]), respectively

$$\begin{aligned} \langle \mu \rangle [j + \frac{1}{2}][i + \frac{1}{2}] &= \left[\frac{1}{4} \left(\mu^{-1}[j][i] + \mu^{-1}[j][i+1] + \mu^{-1}[j+1][i+1] + \mu^{-1}[j+1][i] \right) \right]^{-1} \\ \rho_x[j][i + \frac{1}{2}] &= \frac{1}{2} (\rho[j][i+1] + \rho[j][i]) \\ \rho_y[j + \frac{1}{2}][i] &= \frac{1}{2} (\rho[j+1][i] + \rho[j][i]) \end{aligned} \quad (2.6)$$

The discretization of the linear stress strain relationship in (2.3) at time step n leads to the following system of equations (for simplicity I skip the time index n):

$$\begin{aligned}
u_{xx}[j][i] &\approx \frac{u_x[j][i + \frac{1}{2}] - u_x[j][i - \frac{1}{2}]}{dh} \\
u_{yy}[j][i] &\approx \frac{u_y[j + \frac{1}{2}][i] - u_y[j - \frac{1}{2}][i]}{dh} \\
u_{yx}[j + \frac{1}{2}][i + \frac{1}{2}] &\approx \frac{u_y[j + \frac{1}{2}][i + 1] - u_y[j + \frac{1}{2}][i]}{dh} \\
u_{xy}[j + \frac{1}{2}][i + \frac{1}{2}] &\approx \frac{u_x[j + 1][i + \frac{1}{2}] - u_x[j][i + \frac{1}{2}]}{dh} \\
\sigma_{xy}[j + \frac{1}{2}][i + \frac{1}{2}] &= \langle \mu \rangle [j + \frac{1}{2}][i + \frac{1}{2}] \left(u_{xy}[j + \frac{1}{2}][i + \frac{1}{2}] + u_{yx}[j + \frac{1}{2}][i + \frac{1}{2}] \right) \\
\sigma_{xx}[j][i] &= \lambda[j][i] * \left(u_{xx}[j][i] + u_{yy}[j][i] \right) + 2 * \langle \mu \rangle [j][i] * u_{xx}[j][i] \\
\sigma_{yy}[j][i] &= \lambda[j][i] * \left(u_{xx}[j][i] + u_{yy}[j][i] \right) + 2 * \langle \mu \rangle [j][i] * u_{yy}[j][i]
\end{aligned} \tag{2.7}$$

The discretization of the momentum equation in (2.3) leads to the following system of equations:

$$\begin{aligned}
utt_x^n[j][i + \frac{1}{2}] &= \left(\sigma_{xx}[j][i + 1] - \sigma_{xx}[j][i] + \sigma_{xy}[j + \frac{1}{2}][i] - \sigma_{xy}[j - \frac{1}{2}][i] \right) \\
utt_y^n[j + \frac{1}{2}][i] &= \left(\sigma_{xy}[j][i + \frac{1}{2}] - \sigma_{xy}[j][i - \frac{1}{2}] + \sigma_{yy}[j + 1][i] - \sigma_{yy}[j][i] \right) \\
u_x^{n+1}[j][i + \frac{1}{2}] &= 2 * u_x^n[j][i + \frac{1}{2}] - u_x^{n-1}[j][i + \frac{1}{2}] + \frac{dt^2}{dh * \rho_x[j][i + \frac{1}{2}]} * utt_x^n[j][i + \frac{1}{2}] \\
u_y^{n+1}[j + \frac{1}{2}][i] &= 2 * u_y^n[j + \frac{1}{2}][i] - u_y^{n-1}[j + \frac{1}{2}][i] + \frac{dt^2}{dh * \rho_y[j + \frac{1}{2}][i]} * utt_y^n[j + \frac{1}{2}][i]
\end{aligned} \tag{2.8}$$

2.2.2 Accuracy of FD operators

The derivation of the FD operators in the last section was a simple replacement of the partial derivatives by finite differences. In the following more systematic approach, the first derivative of a variable f at a grid point i is estimated by a Taylor series expansion (Jastram [1992]):

$$\begin{aligned}
(2k-1) \frac{\partial f}{\partial x} \Big|_i &= \frac{1}{dh} (f_{i+(k-1/2)} - f_{i-(k-1/2)}) \\
&+ \frac{1}{dh} \sum_{l=2}^N \frac{((k - \frac{1}{2})dh)^{2l-1}}{(2l-1)!} \frac{\partial^{(2l-1)} f}{\partial x^{(2l-1)}} \Big|_i + \mathcal{O}(dh)^{2N}
\end{aligned}$$

For an operator with length 2N, N equations are added with a weight β_k :

$$\begin{aligned}
\left(\sum_{k=1}^N \beta_k (2k-1) \right) \frac{\partial f}{\partial x} \Big|_i &= \frac{1}{dh} \sum_{k=1}^N \beta_k (f_{i+(k-1/2)} - f_{i-(k-1/2)}) \\
&+ \frac{1}{dh} \sum_{k=1}^N \sum_{l=2}^N \beta_k \frac{((k - \frac{1}{2})dh)^{2l-1}}{(2l-1)!} \frac{\partial^{(2l-1)} f}{\partial x^{(2l-1)}} \Big|_i + \mathcal{O}(dh)^{2N}
\end{aligned} \tag{2.9}$$

The case N=1 leads to the FD operator derived in the last section, which has a length of 2N=2. The Taylor series is truncated after the first term ($\mathcal{O}(dh)^2$). Therefore this operator is called **2nd order FD operator** which refers to the truncation error of the Taylor series and not to the order of the approximated derivative. To understand equation (2.9)

better, we estimate a **4th order FD operator**. This operator has the length $2N = 4$ or $N=2$. The sums in Eq. (2.9) lead to:

$$(\beta_1 + 3\beta_2) \frac{\partial f}{\partial x} \Big|_i = \frac{1}{dh} (\beta_1 (f_{i+1/2} - f_{i-1/2}) + \beta_2 (f_{i+3/2} - f_{i-3/2})) + \frac{dh^3}{dh} \left[\beta_1 \frac{1}{8 \cdot 3!} + \beta_2 \frac{27}{8 \cdot 3!} \right] \frac{\partial^3 f}{\partial x^3} \Big|_i \quad (2.10)$$

The weights β_k can be calculated by the following approach: The factor in front of the partial derivative on the LHS of Eq. (2.10) should equal 1, therefore

$$(\beta_1 + 3\beta_2) = 1.$$

The coefficients in front of $\frac{\partial^3 f}{\partial x^3} \Big|_i$ on the RHS of Eq. (2.10) should vanish:

$$(\beta_1 + 27\beta_2) = 0.$$

The weights β_k can be estimated by solving the matrix equation:

$$\begin{pmatrix} 1 & 3 \\ 1 & 27 \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The resulting coefficients are $\beta_1 = 9/8$ and $\beta_2 = -1/24$. Therefore the 4th order backward- and forward operators are:

$$\begin{aligned} \frac{\partial f}{\partial x} \Big|_{i+1/2} &= \frac{1}{dh} [\beta_1 (f_{i+1} - f_i) + \beta_2 (f_{i+2} - f_{i-1})] && \text{forward operator} \\ \frac{\partial f}{\partial x} \Big|_{i-1/2} &= \frac{1}{dh} [\beta_1 (f_i - f_{i-1}) + \beta_2 (f_{i+1} - f_{i-2})] && \text{backward operator} \end{aligned} \quad (2.11)$$

The coefficients β_i in the FD operator are called **Taylor coefficients**. The accuracy of higher order FD operators can be improved by seeking for FD coefficients β_k that approximate the first derivative in a certain frequency range (Holberg [1987]). These numerically optimized coefficients are called **Holberg coefficients**.

2.2.3 Initial and Boundary Conditions

To find a unique solution of the problem, initial and boundary conditions have to be defined. The initial conditions for the elastic forward problem are:

$$\begin{aligned} u_i(\mathbf{x}, t) &= 0 \\ \frac{\partial u_i(\mathbf{x}, t)}{\partial t} &= 0 \end{aligned} \quad (2.12)$$

for all $\mathbf{x} \in V$ at $t = 0$.

For the geophysical application two types of boundary conditions are very important:

1. **Horizontal Free Surface:** The interface between the elastic medium and air at the surface is very important when trying to model surface waves or multiple reflections in a marine environment. Since all stresses in the normal direction at this interface vanish

$$\sigma_{xy} = \sigma_{yy} = 0.0 \quad (2.13)$$

this boundary condition is called (stress) **free surface**. Two types of implementations are common. In the implicit definition of the free surface, a small layer with the acoustic parameters of air ($V_p = 300$ m/s, $V_s = 0.0$ m/s, $\rho = 1.25$ kg/m³) is placed on top of the model. One advantage of the implicit definition of the free surface is the easy implementation of topography on the FD grid, however to get accurate results for surface waves or multiples, this approach requires a fine spatial sampling of the FD grid near the free surface. An explicit free surface can be implemented by using the mirroring technique by Levander, which leads to stable and accurate

solutions for plain interfaces (Levander [1988], Robertsson et al. [1995]). If the planar free surface is located at grid point $j = h$, the stress at this point is set to zero and the stresses below the free surface are mirrored with an inverse sign:

$$\begin{aligned}\sigma_{yy}(h, i) &= 0 \\ \sigma_{yy}(h-1, i) &= -\sigma_{yy}(h+1, i) \\ \sigma_{xy}(h-\frac{1}{2}, i+\frac{1}{2}) &= -\sigma_{xy}(h+\frac{1}{2}, i+\frac{1}{2}) \\ \sigma_{xy}(h-\frac{3}{2}, i+\frac{1}{2}) &= -\sigma_{xy}(h+\frac{3}{2}, i+\frac{1}{2})\end{aligned}\tag{2.14}$$

When updating the stress component $\sigma_{xx} = (\lambda + 2\mu)u_{xx} + \lambda u_{yy}$ at the free surface, only horizontal particle displacements should be used because vertical derivatives over the free surface lead to instabilities (Levander [1988]). The vertical derivative of the y-displacement u_{yy} can be replaced by using the boundary condition at the free surface:

$$\begin{aligned}\sigma_{yy} &= (\lambda + 2\mu)u_{yy} + \lambda u_{xx} = 0 \\ u_{yy} &= -\frac{\lambda}{(\lambda + 2\mu)}u_{xx}\end{aligned}\tag{2.15}$$

Therefore the stress σ_{xx} can be written as

$$\sigma_{xx} = \frac{4(\lambda\mu + \mu^2)}{\lambda + 2\mu}u_{xx}\tag{2.16}$$

2. Absorbing Boundary Conditions: Due to limited computational resources, the FD grid has to be as small as possible. To model problems with an infinite extension in different directions, e.g. a full or half-space problem, an artificial absorbing boundary condition has to be applied. A very effective way to damp the waves near the boundaries are **Perfectly Matched Layers (PMLs)**. This can be achieved by a coordinate stretch of the wave equations in the frequency domain (Komatitsch and Martin [2007]). The coordinate stretch creates exponentially decaying plane wave solutions in the absorbing boundary frame. The PML's are only reflectionless if the exact wave equation is solved. As soon as the problem is discretized (for example using finite differences) you are solving an approximate wave equation and the analytical perfection of the PML is no longer valid. To overcome this shortcoming the wavefield is damped by the damping function

$$c = -V_{pml} * \frac{\log(\alpha)}{L}\tag{2.17}$$

where V_{pml} denotes the typical P-wave velocity of the medium in the absorbing boundary frame, $\alpha = 1 \times 10^{-4}$ and L is the thickness of the absorbing boundary layer. A comparison between the exponential damping and the PML boundary is shown in Fig.2.2. The PMLs are damping the seismic waves by a factor 5-10 more effective than the absorbing boundary frame.

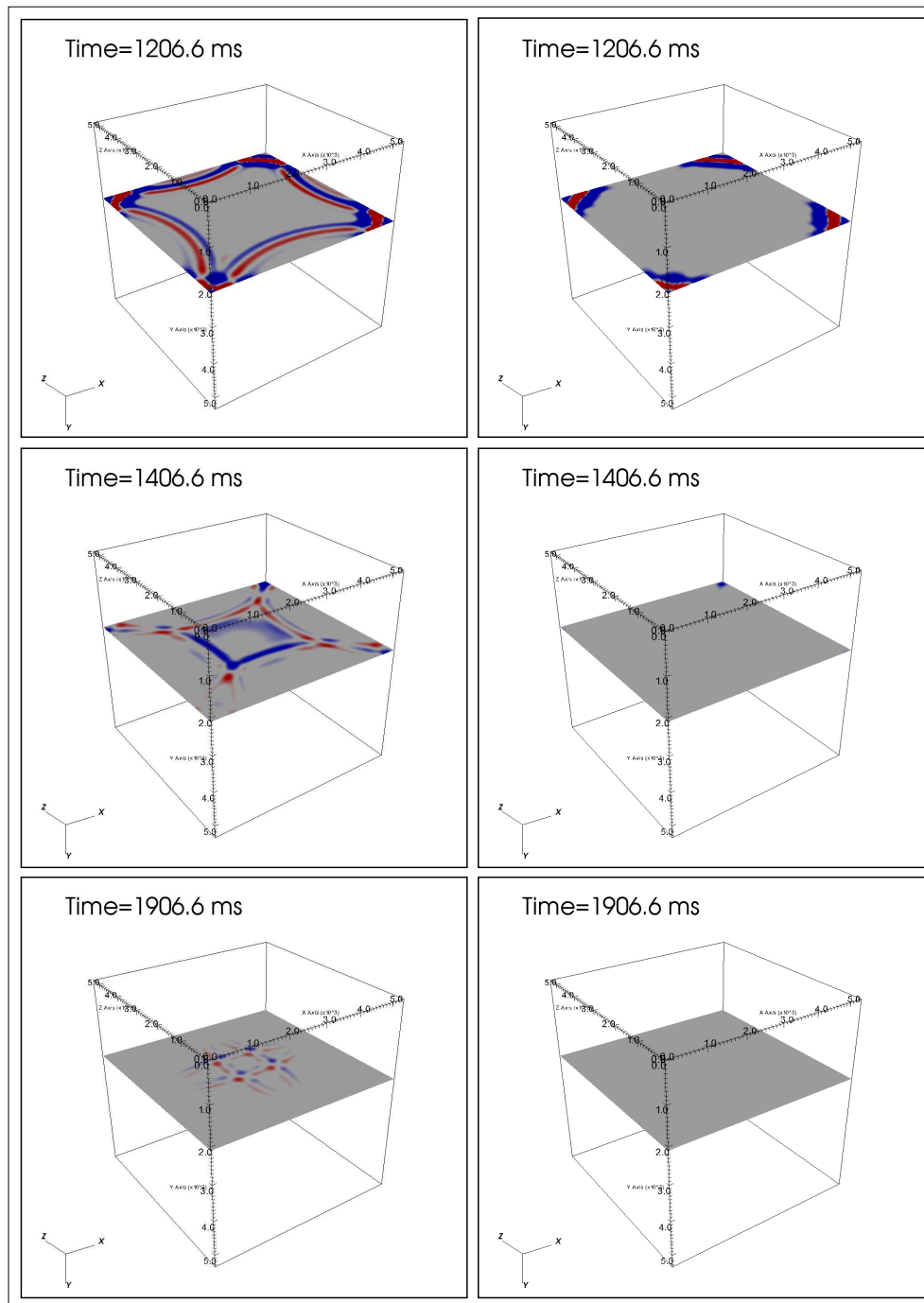


Figure 2.2: Comparison between exponential damping (left column) and PML (right column) absorbing boundary conditions for a homogeneous full space model.

2.3 Numerical Artefacts and Instabilities

To avoid numerical artefacts and instabilities during a FD modelling run, spatial and temporal sampling conditions for the wavefield have to be satisfied. These will be discussed in the following two sections.

2.3.1 Grid Dispersion

The first question when building a FD model is: What is the maximum spatial grid point distance dh , for a correct sampling of the wavefield ? To answer this question we take a look at this simple example: The particle displacement in x-direction is defined by a sine function:

$$u_x = \sin\left(2\pi \frac{x}{\lambda}\right), \quad (2.18)$$

where λ denotes the wavelength. When calculating the derivation of this function analytically at $x = 0$ and setting $\lambda = 1$ m we get:

$$\left. \frac{du_x}{dx} \right|_{x=0} = \frac{2\pi}{\lambda} \cos\left(2\pi \frac{x}{\lambda}\right) \Big|_{x=0} = 2\pi. \quad (2.19)$$

In the next step the derivation is approximated numerically by a staggered 2nd order finite-difference operator:

$$\left. \frac{du_x}{dx} \right|_{x=0} \approx \frac{u_x(x + \frac{1}{2}\Delta x) - u_x(x - \frac{1}{2}\Delta x)}{\Delta x} \Big|_{x=0} = \frac{\sin\left(\frac{2\pi(x + \frac{1}{2}\Delta x)}{\lambda}\right) - \sin\left(\frac{2\pi(x - \frac{1}{2}\Delta x)}{\lambda}\right)}{\Delta x}. \quad (2.20)$$

Using the Nyquist-Shannon sampling theorem it should be sufficient to sample the wavefield with $\Delta x = \lambda/2$. In table 2.1 the numerical solutions of eq. (2.20) and the analytical solution (2.19) are compared for different sample intervals $\Delta x = \lambda/n$, where n is the number of gridpoints per wavelength. For the case $n=2$, which corresponds to the Nyquist-Shannon theorem, the numerical solution is $\left. \frac{du_x}{dx} \right|_{x=0} = 4.0$, which is not equal with the analytical solution 2π . A refinement of the spatial sampling of the wavefield results in an improvement of the finite difference solution. For $n = 16$ the numerical solution is accurate to the second decimal place. The effect of a sparsely sampled pressure field is illustrated in figure 2.3 for a homogeneous block model with stress free surfaces. The dimensions of the FD grid are fixed and the central frequency of the source signal is increased systematically. When using a spatial sampling of 16 grid points per minimum wavelength (figure 2.3, top) the wavefronts are sharply defined. For $n = 4$ grid points a slight numerical dispersion of the wave occurs (figure 2.3, center). This effect is obvious when using the Nyquist criterion ($n = 2$) (figure 2.3, bottom). Since the numerical calculated wavefield seem to be dispersive this numerical artefact is called **grid dispersion**. To avoid the occurrence of grid dispersion the following criteria for the spatial grid spacing dh has to be satisfied:

$$dh \leq \frac{\lambda_{\min}}{n} = \frac{V_{\min}}{n f_{\max}}. \quad (2.21)$$

Here $\lambda_{\min}$ denotes the minimum wavelength, $V_{\min}$ the minimum velocity in the model and $f_{\max}$ is the maximum frequency of the source signal. Depending on the accuracy of the used FD operator the parameter n is different. In table 2.2 n is listed for different FD operator lengths and types (Taylor and Holberg operators). The Holberg coefficients are calculated for a minimum dispersion error of 0.1% at $3f_{\max}$. For short operators n should be choosen relatively large, so the spatial grid spacing is small, while for longer FD operators n is smaller and the grid spacing can be larger.

n	Δx [m]	$\frac{dv_x}{dx} _{x=0}$ []
analytical	-	$2\pi \approx 6.283$
2	$\lambda/2$	4.0
4	$\lambda/4$	5.657
8	$\lambda/8$	6.123
16	$\lambda/16$	6.2429
32	$\lambda/32$	6.2731

Table 2.1: Comparison of the analytical solution Eq. (2.19) with the numerical solution Eq. (2.20) for different grid spacings $\Delta x = \lambda/n$.

FDORDER	n (Taylor)	n (Holberg)
2nd	12	12
4th	8	8.32
6th	6	4.77
8th	5	3.69
10th	5	3.19
12th	4	2.91

Table 2.2: The number of grid points per minimum wavelength n for different orders (2nd-12th) and types (Taylor and Holberg) of FD operators. For the Holberg coefficients n is calculated for a minimum dispersion error of 0.1% at $3f_{\max}$.

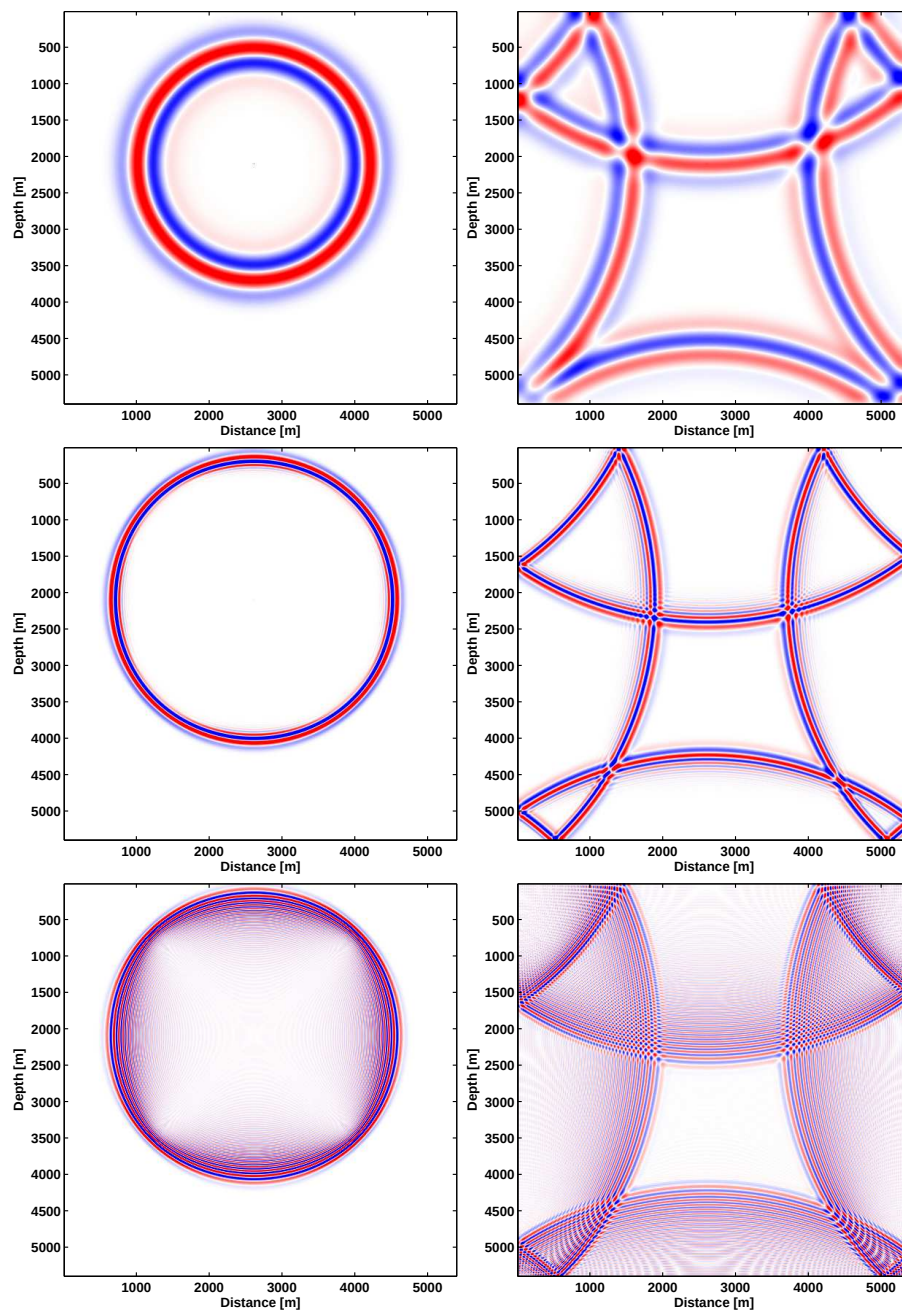


Figure 2.3: The influence of grid dispersion in FD modeling: Spatial sampling of the wavefield using $n=16$ (top), $n=4$ (center) and $n=2$ gridpoints (bottom) per minimum wavelength $\lambda_{\min}$.

2.3.2 The Courant Instability

Beside the spatial, the temporal discretization has to satisfy a sampling criterion to ensure the stability of the FD code. If a wave is propagating on a discrete grid, then the timestep dt has to be less than the time for the wave to travel between two adjacent grid points with grid spacing dh . For an elastic 2D grid this means mathematically:

$$dt \leq \frac{dh}{h\sqrt{2}V_{\max}}, \quad (2.22)$$

where $V_{\max}$ is the maximum velocity in the model. The factor h depends on the order of the FD operator and can easily be calculated by summing over the weighting coefficients β_i

$$h = \sum_i \beta_i. \quad (2.23)$$

In table 2.3 h is listed for different FD operator lengths and types (Taylor and Holberg operators). Criterion (2.22) is called **Courant-Friedrichs-Lewy criterion** (Courant et al. [1928], Courant et al. [March 1967]). figure 2.4 shows the evolution of the pressure field when the Courant criterion is violated. After a few time steps the amplitudes are growing to infinity and the calculation becomes unstable.

FDORDER	h (Taylor)	h (Holberg)
2nd	1.0	1.0
4th	7/6	1.184614
6th	149/120	1.283482
8th	2161/1680	1.345927
10th	53089/40320	1.387660
12th	1187803/887040	1.417065

Table 2.3: The factor h in the Courant criterion for different orders (2nd-12th) and types (Taylor and Holberg) of FD operators.

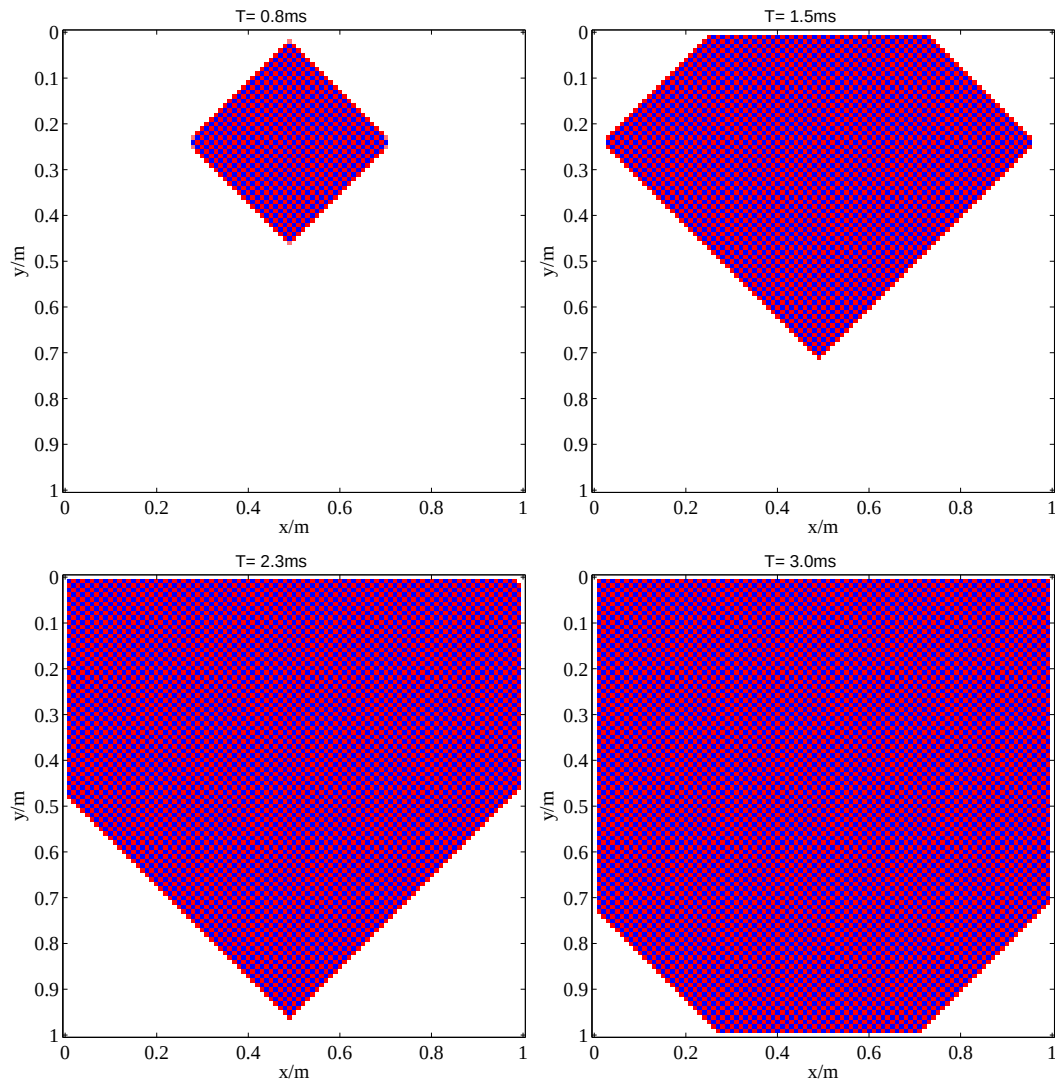


Figure 2.4: Temporal evolution of the Courant instability. In the colored areas the wave amplitudes are extremely large.

Chapter 3

The adjoint problem

The aim of full waveform tomography is to find an "optimum" model which can explain the data very well. It should not only explain the first arrivals of specific phases of the seismic wavefield like refractions or reflections, but also the amplitudes which contain information on the distribution of the elastic material parameters in the underground. To achieve this goal three problems have to be solved:

1. What is an "optimum" model ?
2. How can this model be found ?
3. Is this model unique or are other models existing, which could explain the data equally well ?

3.1 What is an "optimum" model ?

In reflection seismics the i^{th} component of the elastic displacement field $u_i(\mathbf{x}_s, \mathbf{x}_r, t)$ excited by sources located at $\mathbf{x}_s$ will be recorded by receivers at $\mathbf{x}_r$ at time t . For a given distribution of the material parameters the forward problem Eq. 2.3 can be solved by finite differences (section 2.2). The result is a model data set $\mathbf{u}^{\text{mod}}$. This modelled data can be compared with the field data $\mathbf{u}^{\text{obs}}$. If the misfit or data residuals $\delta\mathbf{u} = \mathbf{u}^{\text{mod}} - \mathbf{u}^{\text{obs}}$ (figure 3.1) between the modelled and the field data is small the model can explain the data very well. If the residuals are large the model cannot explain the data. The misfit can be measured by a vector norm $|\mathbf{L}|_p$ which is defined for $p = 1, 2, \dots$ as

$$|\mathbf{L}|_p = \left(\sum_i |\delta u_i|^p \right)^{1/p} \quad (3.1)$$

The special case $|\mathbf{L}|_\infty$ is defined as

$$|\mathbf{L}|_\infty = \max_i |\delta u_i| \quad (3.2)$$

The L2-norm

$$E = |\mathbf{L}|_2 = \frac{1}{2} \delta \mathbf{u}^T \delta \mathbf{u} \quad (3.3)$$

has a special physical meaning. It represents the residual elastic energy contained in the data residuals $\delta\mathbf{u}$. An optimum model can be found in a minimum of the residual energy. Therefore the optimum model is the solution of a nonlinear optimization problem.

3.2 How to find an optimum model

Figure 3.2 shows a schematic sketch of the residual energy at one point in space as a function of two model parameters λ and μ . The colors represent different values of the residual energy. Red areas represent models with high residual energy which do not fit the data, while the blue parts are good fitting models with low residual energies. The aim is to find the minimum of the residual energy marked by the red cross. Starting at a point $\mathbf{m}_1 = (\lambda_1(\mathbf{x}), \mu_1(\mathbf{x}), \rho_1(\mathbf{x}), \dots)$ in the parameter space we want to find the minimum by updating the material parameters in an iterative way

$$\mathbf{m}_2 = \mathbf{m}_1 + \mu_1 \delta \mathbf{m}_1, \quad (3.4)$$

along the search direction $\delta \mathbf{m}_1$ with the step length μ_1 . To find the optimum search direction $\delta \mathbf{m}_1$ we expand the residual energy $E(\mathbf{m}_1 + \delta \mathbf{m}_1)$ near the starting point in a Taylor series:

$$E(\mathbf{m}_1 + \delta \mathbf{m}_1) \approx E(\mathbf{m}_1) + \delta \mathbf{m}_1 \left(\frac{\partial E}{\partial \mathbf{m}} \right)_1 + \frac{1}{2} \delta \mathbf{m}_1 \left(\frac{\partial^2 E}{\partial \mathbf{m}^2} \right)_1 \delta \mathbf{m}_1^T \quad (3.5)$$

and set the derivative of Eq. 3.5 with respect to $\delta \mathbf{m}_1$ zero

$$\frac{\partial E(\mathbf{m}_1 + \delta \mathbf{m}_1)}{\partial \delta \mathbf{m}_1} = \left(\frac{\partial E}{\partial \mathbf{m}} \right)_1 + \delta \mathbf{m}_1 \left(\frac{\partial^2 E}{\partial \mathbf{m}^2} \right)_1 = 0 \quad (3.6)$$

Which finally leads to

$$\delta \mathbf{m}_1 = - \left(\frac{\partial^2 E}{\partial \mathbf{m}^2} \right)_1^{-1} \left(\frac{\partial E}{\partial \mathbf{m}} \right)_1 = -\mathbf{H}_1^{-1} \left(\frac{\partial E}{\partial \mathbf{m}} \right)_1 \quad (3.7)$$

where $(\partial E / \partial \mathbf{m})_1$ denotes the steepest-descent direction of the objective function and $\mathbf{H}_1^{-1}$ the inverse Hessian matrix. The inverse Hessian matrix for the elastic problem is often singular and can only be calculated with high computational costs. Therefore the inverse Hessian matrix is approximated by a preconditioning operator $\mathbf{P}$. There is

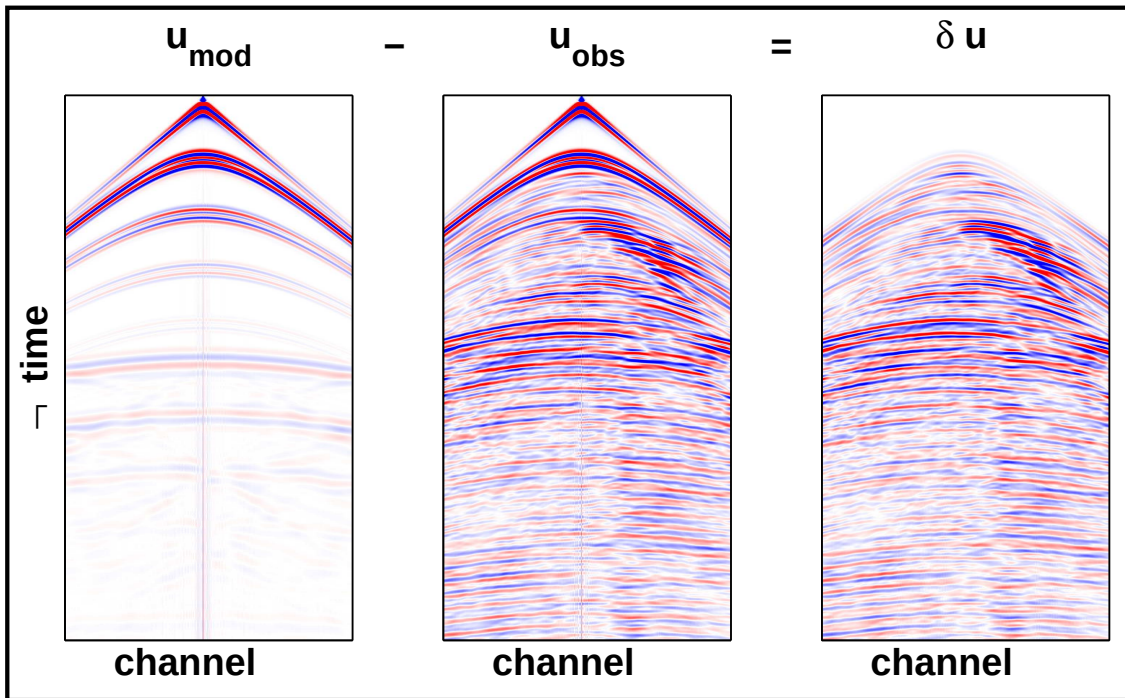


Figure 3.1: Definition of data residuals $\delta \mathbf{u}$.

no general rule for an optimum preconditioning operator, but two very simple operators are described in more detail in chapter ?? for a cross-well acquisition geometry and in chapter 7.1.3 for a reflection geometry.

$$\delta \mathbf{m}_1 \approx -\mathbf{P}_1 \left(\frac{\partial E}{\partial \mathbf{m}} \right)_1. \quad (3.8)$$

By replacing $\delta \mathbf{m}_1$ in Eq. 3.4 with Eq. 3.8 we get

$$\mathbf{m}_2 = \mathbf{m}_1 - \mu_1 \mathbf{P}_1 \left(\frac{\partial E}{\partial \mathbf{m}} \right)_1, \quad (3.9)$$

The optimum model parameters can be found along the negative gradient direction of the residual energy. The starting point $\mathbf{m}_1$ is not a particular point, so the update function can be applied to every point in the parameter space $\mathbf{m}_n$

$$\mathbf{m}_{n+1} = \mathbf{m}_n - \mu_n \mathbf{P}_n \left(\frac{\partial E}{\partial \mathbf{m}} \right)_n. \quad (3.10)$$

3.3 Calculation of the gradient direction $\frac{\partial E}{\partial \mathbf{m}}$

To estimate the gradient direction $\partial E / \partial \mathbf{m}$ the residual energy is rewritten as:

$$E = \frac{1}{2} \delta \mathbf{u}^T \delta \mathbf{u} = \frac{1}{2} \sum_{\text{sources}} \int dt \sum_{\text{receiver}} \delta \mathbf{u}^2(\mathbf{x}_r, \mathbf{x}_s, t) \quad (3.11)$$

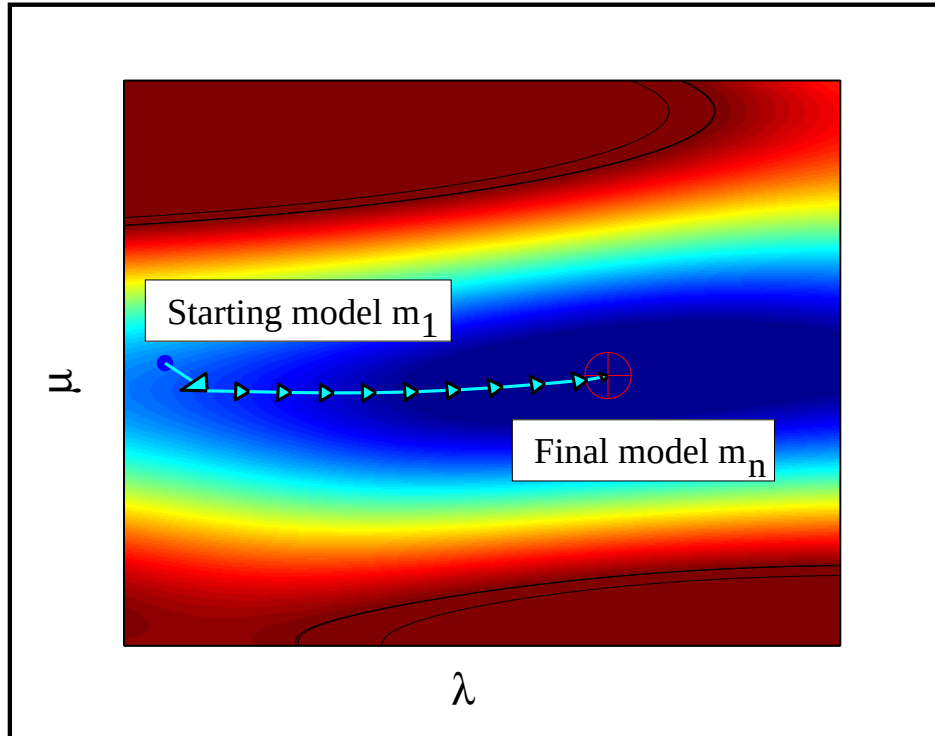


Figure 3.2: Schematic sketch of the residual energy at one point in space as a function of two model parameters m_1 and m_2 . The blue dot denotes the starting point in the parameter space, while the red cross marks a minimum of the objective function.

After derivation with respect to a model parameter $\mathbf{m}$ we get

$$\begin{aligned}
 \frac{\partial E}{\partial \mathbf{m}} &= \sum_{\text{sources}} \int dt \sum_{\text{receiver}} \frac{\partial \delta \mathbf{u}}{\partial \mathbf{m}} \delta \mathbf{u} \\
 &= \sum_{\text{sources}} \int dt \sum_{\text{receiver}} \frac{\partial (\mathbf{u}^{\text{mod}}(\mathbf{m}) - \mathbf{u}^{\text{obs}})}{\partial \mathbf{m}} \delta \mathbf{u} \\
 &= \sum_{\text{sources}} \int dt \sum_{\text{receiver}} \frac{\partial \mathbf{u}^{\text{mod}}(\mathbf{m})}{\partial \mathbf{m}} \delta \mathbf{u}
 \end{aligned} \tag{3.12}$$

Eq. (3.12) can be related to the mapping of small changes from the data to the model space and vice versa (figure 3.3). A small change in the model space $\delta \mathbf{m}$, e.g. one model parameter at one point in space, will result in a small

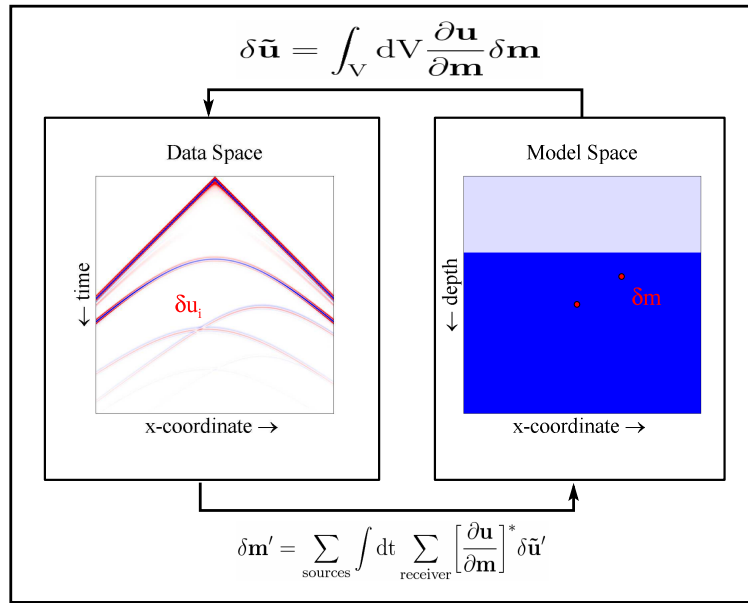


Figure 3.3: Mapping between model and data space and vice versa.

perturbation of the data space $\delta \tilde{\mathbf{u}}$, e.g. one wiggle in the seismic section. If the Frechét derivative $\frac{\partial \mathbf{u}}{\partial \mathbf{m}}$ is known, all the small perturbations in model space can be integrated over the model volume V to calculate the total change in data space (Tarantola [2005]):

$$\delta \tilde{\mathbf{u}}(\mathbf{x}_s, \mathbf{x}_r, t) = \int_V dV \frac{\partial \mathbf{u}}{\partial \mathbf{m}} \delta \mathbf{m}, \tag{3.13}$$

or by introducing the linear operator $\hat{L}$

$$\delta \tilde{\mathbf{u}} = \hat{L} \delta \mathbf{m} := \int_V dV \frac{\partial \mathbf{u}}{\partial \mathbf{m}} \delta \mathbf{m}.$$

In a similar way small changes in the data space $\delta \tilde{\mathbf{u}}'$ can be integrated to calculate the total change in the model space $\delta \mathbf{m}'$ (Tarantola [2005])

$$\delta \mathbf{m}' = \sum_{\text{sources}} \int dt \sum_{\text{receiver}} \left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}} \right]^* \delta \tilde{\mathbf{u}}', \tag{3.14}$$

or as operator equation

$$\delta \mathbf{m}' = \hat{L}^* \delta \tilde{\mathbf{u}}'.$$

In this case the Frechét derivative $\frac{\partial \mathbf{u}}{\partial \mathbf{m}}$ is replaced by its adjoint counterpart $\frac{\partial \mathbf{u}}{\partial \mathbf{m}}^*$. Note that $\delta \tilde{\mathbf{u}} \neq \delta \tilde{\mathbf{u}}'$ and $\delta \mathbf{m} \neq \delta \mathbf{m}'$, so there is no unique way to map perturbations from the model to the data space or vice versa. Because the operator $\hat{\mathbf{L}}$ is linear, the kernel of $\hat{\mathbf{L}}$ and its adjoint counterpart $\hat{\mathbf{L}}^*$ are identical (see chapter 5.4.2 in Tarantola [2005])

$$\left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}} \right]^* = \left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}} \right]$$

Therefore the mapping from the data to the model space Eq. (3.14) is equal to the gradient of the residual energy Eq. (3.12):

$$\begin{aligned} \delta \mathbf{m}' &= \sum_{\text{sources}} \int dt \sum_{\text{receiver}} \left[\frac{\partial \mathbf{u}_i}{\partial \mathbf{m}} \right]^* \delta \tilde{\mathbf{u}}' \\ &= \sum_{\text{sources}} \int dt \sum_{\text{receiver}} \left[\frac{\partial \mathbf{u}_i}{\partial \mathbf{m}} \right] \delta \mathbf{u} \\ &= \frac{\partial E}{\partial \mathbf{m}} \end{aligned} \quad (3.15)$$

if the perturbation of the data space $\delta \tilde{\mathbf{u}}'$ is interpreted as data residuals $\delta \mathbf{u}$. So the approach to estimate the gradient direction $\partial E / \partial \mathbf{m}$ can be split into 3 parts

1. Find a solution to the forward problem

$$\delta \mathbf{u} = \hat{\mathbf{L}} \delta \mathbf{m}.$$

2. Identify the Frechét kernels $\partial \mathbf{u} / \partial \mathbf{m}$

3. Use the property, that a linear operator $\hat{\mathbf{L}}$ and its adjoint $\hat{\mathbf{L}}^*$ have the same kernels and calculate the gradient direction by using:

$$\frac{\partial E}{\partial \mathbf{m}} = \delta \mathbf{m}' = \hat{\mathbf{L}}^* \delta \mathbf{u}'.$$

This is a very general approach. Now we apply this approach to the equations of motion for an elastic medium. The following derivation is much easier, when assuming a general elastic medium first and introduce the isotropy later on. Therefore the elastic forward problem Eqs. (2.3) can be written as

$$\begin{aligned} \rho \frac{\partial^2 u_i}{\partial t^2} - \frac{\partial}{\partial x_j} \sigma_{ij} &= f_i, \\ \sigma_{ij} - c_{ijkl} \epsilon_{kl} &= T_{ij}, \\ \epsilon_{ij} &= \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \\ &+ \text{initial and boundary conditions,} \end{aligned} \quad (3.16)$$

where ρ denotes the density, σ_{ij} the stress tensor, ϵ_{ij} the strain tensor, c_{ijkl} the stiffness tensor, f_i , T_{ij} source terms for volume and surface forces, respectively. In the next step every parameter and variable in the elastic wave equation is perturbed by a first order perturbation as shown in Fig. 3.3:

$$\begin{aligned} u_i &\rightarrow u_i + \delta u_i, \\ \rho &\rightarrow \rho + \delta \rho, \\ \sigma_{ij} &\rightarrow \sigma_{ij} + \delta \sigma_{ij}, \\ c_{ijkl} &\rightarrow c_{ijkl} + \delta c_{ijkl}, \\ \epsilon_{ij} &\rightarrow \epsilon_{ij} + \delta \epsilon_{ij}, \end{aligned} \quad (3.17)$$

These substitutions yield new equations of motion describing the displacement perturbations δu_i and stress perturbations $\delta \sigma_{ij}$ as a function of new source terms Δf_i and ΔT_{ij}

$$\begin{aligned} \rho \frac{\partial^2 \delta u_i}{\partial t^2} - \frac{\partial}{\partial x_j} \delta \sigma_{ij} &= \Delta f_i, \\ \delta \sigma_{ij} - c_{ijkl} \delta \epsilon_{kl} &= \Delta T_{ij}, \\ \delta \epsilon_{ij} &= \frac{1}{2} \left(\frac{\partial \delta u_i}{\partial x_j} + \frac{\partial \delta u_j}{\partial x_i} \right) \\ &+ \text{perturbated initial and boundary conditions} \end{aligned} \quad (3.18)$$

The new source terms are

$$\Delta f_i = -\delta \rho \frac{\partial^2 u_i}{\partial t^2} \quad (3.19)$$

and

$$\Delta T_{ij} = \delta c_{ijkl} \epsilon_{kl}. \quad (3.20)$$

Two points are important to notice:

1. Eq.(3.18) states that every change of a material parameter acts as a source (Eq.(3.19) and Eq.(3.20)), but the perturbed wavefield is propagating in the unperturbed medium.
2. The new wave equation (3.18) has mathematically the same form as the unperturbed elastic wave equation, and hence its solution can be obtained in terms of Green's functions G_{ij} of the elastic wave equation.

The solution of the perturbed elastic equations of motion (3.18) in terms of the elastic Green's function $G_{ij}(\mathbf{x}, t; \mathbf{x}', t')$ can be written as:

$$\begin{aligned} \delta u_i(\mathbf{x}, t) &= \int_V dV \int_0^T dt' G_{ij}(\mathbf{x}, t; \mathbf{x}', t') \Delta f_j(\mathbf{x}', t') \\ &- \int_V dV \int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(\mathbf{x}, t; \mathbf{x}', t') \Delta T_{jk}(\mathbf{x}', t'). \end{aligned} \quad (3.21)$$

Substituting the force and traction terms given by Eqs.(3.19) and (3.20) into Eq.(3.21) yields after some rearranging

$$\begin{aligned} \delta u_i(\mathbf{x}, t) &= - \int_V dV \int_0^T dt' G_{ij}(\mathbf{x}, t; \mathbf{x}', t') \frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}', t') \delta \rho \\ &- \int_V dV \int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(\mathbf{x}, t; \mathbf{x}', t') \epsilon_{lm}(\mathbf{x}', t') \delta c_{ijklm} \end{aligned} \quad (3.22)$$

Introducing isotropy via (Aki and Richards [1980], p.23)

$$\delta c_{ijklm} = \delta_{jk} \delta_{lm} \delta \lambda + (\delta_{jl} \delta_{km} + \delta_{jm} \delta_{kl}) \delta \mu. \quad (3.23)$$

leads to:

$$\begin{aligned} \delta u_i(\mathbf{x}, t) &= - \int_V dV \left[\int_0^T dt' G_{ij}(\mathbf{x}, t; \mathbf{x}', t') \frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}', t') \right] \delta \rho \\ &- \int_V dV \left[\int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(\mathbf{x}, t; \mathbf{x}', t') \epsilon_{lm}(\mathbf{x}', t') \delta_{jk} \delta_{lm} \right] \delta \lambda \\ &- \int_V dV \left[\int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(\mathbf{x}, t; \mathbf{x}', t') \epsilon_{lm}(\mathbf{x}', t') (\delta_{jl} \delta_{km} + \delta_{jm} \delta_{kl}) \right] \delta \mu. \end{aligned} \quad (3.24)$$

Utilization of Eq.(3.24) to solve the forward problem is known as the Born approximation. In waveform tomography the Born approximation is not used to solve the forward problem. Instead the full elastic wave equation is solved. Equation (3.24) has the same form as the desired expression for the forward problem Eqs.(3.13):

$$\delta \mathbf{u} = \int_V dV \frac{\partial \mathbf{u}}{\partial \mathbf{m}} \delta \mathbf{m}. \quad (3.25)$$

Therefore the Frechét kernels $\frac{\partial u_i}{\partial \mathbf{m}(\mathbf{x})}$ for the individual material parameters can be identified as:

$$\begin{aligned} \frac{\partial u_i}{\partial \rho} &= - \int_0^T dt' G_{ij}(\mathbf{x}, t; \mathbf{x}', t') \frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}', t') \\ \frac{\partial u_i}{\partial \lambda} &= - \int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(\mathbf{x}, t; \mathbf{x}', t') \epsilon_{lm}(\mathbf{x}', t') \delta_{jk} \delta_{lm} \\ \frac{\partial u_i}{\partial \mu} &= - \int_0^T dt' \frac{\partial G_{ij}}{\partial x'_k}(\mathbf{x}, t; \mathbf{x}', t') \epsilon_{lm}(\mathbf{x}', t') (\delta_{jl} \delta_{km} + \delta_{jm} \delta_{kl}) \end{aligned} \quad (3.26)$$

By definition the adjoint of the operator (3.25) can be written as

$$\delta \mathbf{m}'(\mathbf{x}) = \sum_{\text{sources}} \int_0^T dt \sum_{\alpha=1}^{N_{\text{rec}}} \left[\frac{\partial u_i}{\partial \mathbf{m}} \right]^* \delta u'_i(\mathbf{x}_\alpha, t'), \quad (3.27)$$

Because a linear operator and its transpose have the same kernels $\partial u_i / \partial \mathbf{m}$, the only difference arise in the variables of sum/integration, which are complementary. Inserting the integral kernels (3.26) in Eq.(3.27) yields

$$\begin{aligned} \delta \rho' &= - \sum_{\text{sources}} \int_0^T dt \sum_{\alpha=1}^{N_{\text{rec}}} \int_0^T dt' G_{ij}(\mathbf{x}_\alpha, t'; \mathbf{x}, t) \frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}, t) \delta u'_i(\mathbf{x}_\alpha, t'), \\ \delta \lambda' &= - \sum_{\text{sources}} \int_0^T dt \sum_{\alpha=1}^{N_{\text{rec}}} \int_0^T dt' \frac{\partial G_{ij}}{\partial x_k}(\mathbf{x}_\alpha, t'; \mathbf{x}, t) \epsilon_{lm}(\mathbf{x}, t) \delta_{jk} \delta_{lm} \delta u'_i(\mathbf{x}_\alpha, t'), \\ \delta \mu' &= - \sum_{\text{sources}} \int_0^T dt \sum_{\alpha=1}^{N_{\text{rec}}} \int_0^T dt' \frac{\partial G_{ij}}{\partial x_k}(\mathbf{x}_\alpha, t'; \mathbf{x}, t) \epsilon_{lm}(\mathbf{x}, t) (\delta_{jl} \delta_{km} + \delta_{jm} \delta_{kl}) \delta u'_i(\mathbf{x}_\alpha, t'). \end{aligned}$$

The terms only depending on time t and the positions $\mathbf{x}$ can be moved infront of the sum over the receivers

$$\begin{aligned} \delta \rho' &= - \sum_{\text{sources}} \int_0^T dt \frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}, t) \sum_{\alpha=1}^{N_{\text{rec}}} \int_0^T dt' G_{ij}(\mathbf{x}_\alpha, t'; \mathbf{x}, t) \delta u'_i(\mathbf{x}_\alpha, t'), \\ \delta \lambda' &= - \sum_{\text{sources}} \int_0^T dt \epsilon_{lm}(\mathbf{x}, t) \delta_{jk} \delta_{lm} \sum_{\alpha=1}^{N_{\text{rec}}} \int_0^T dt' \frac{\partial G_{ij}}{\partial x_k}(\mathbf{x}_\alpha, t'; \mathbf{x}, t) \delta u'_i(\mathbf{x}_\alpha, t'), \\ \delta \mu' &= - \sum_{\text{sources}} \int_0^T dt \epsilon_{lm}(\mathbf{x}, t) (\delta_{jl} \delta_{km} + \delta_{jm} \delta_{kl}) \sum_{\alpha=1}^{N_{\text{rec}}} \int_0^T dt' \frac{\partial G_{ij}}{\partial x_k}(\mathbf{x}_\alpha, t'; \mathbf{x}, t) \delta u'_i(\mathbf{x}_\alpha, t'). \end{aligned} \quad (3.28)$$

Defining the wavefield

$$\Psi_j(\mathbf{x}, t) = \sum_{\alpha=1}^{N_{\text{rec}}} \int_0^T dt' G_{ij}(\mathbf{x}_\alpha, t'; \mathbf{x}, t) \delta u'_i(\mathbf{x}_\alpha, t'), \quad (3.29)$$

Eqs.(3.28) can be written as

$$\begin{aligned}\delta\rho' &= - \sum_{\text{sources}} \int_0^T dt \frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}, t) \Psi_j, \\ \delta\lambda' &= - \sum_{\text{sources}} \int_0^T dt \epsilon_{lm}(\mathbf{x}, t) \delta_{jk} \delta_{lm} \frac{\partial \Psi_j}{\partial x_k}, \\ \delta\mu' &= - \sum_{\text{sources}} \int_0^T dt \epsilon_{lm}(\mathbf{x}, t) (\delta_{jl} \delta_{km} + \delta_{jm} \delta_{kl}) \frac{\partial \Psi_j}{\partial x_k}.\end{aligned}\tag{3.30}$$

The wavefield Ψ_j is generated by propagating the residual data $\delta u'_i$ from the receiver positions backwards in time through the elastic medium. To obtain a more symmetric expression for the density gradient, let us integrate the density gradient in (3.30) by parts

$$\begin{aligned}\delta\rho' &= - \sum_{\text{sources}} \int_0^T dt \left(\frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}, t) \right) \Psi_j(\mathbf{x}, t) \\ &= - \sum_{\text{sources}} \left\{ \left[\frac{\partial u_j}{\partial t}(\mathbf{x}, T) \Psi_j(\mathbf{x}, T) \right]_0^T - \int_0^T dt \frac{\partial u_j}{\partial t}(\mathbf{x}, t) \frac{\partial \Psi_j}{\partial t}(\mathbf{x}, t) \right\}.\end{aligned}\tag{3.31}$$

According to Eqs. (2.12) the field $u_j(\mathbf{x}, t)$ satisfies initial conditions of rest, $u_j(\mathbf{x}, 0) = 0$ and $\partial u_j(\mathbf{x}, 0)/\partial t = 0$. The field $\Psi_j(\mathbf{x}, t)$ satisfies final conditions of rest, $\Psi_j(\mathbf{x}, T) = 0$. Therefore

$$\delta\rho' = - \sum_{\text{sources}} \int_0^T dt \left(\frac{\partial^2 u_j}{\partial t^2}(\mathbf{x}, t) \right) \Psi_j(\mathbf{x}, t) = \sum_{\text{sources}} \int_0^T dt \frac{\partial u_j}{\partial t}(\mathbf{x}, t) \frac{\partial \Psi_j}{\partial t}(\mathbf{x}, t).\tag{3.32}$$

Writing out the implicit sums in the gradients of the Lamé parameters $\delta\lambda'$ and $\delta\mu'$ in Eqs. (3.30)

$$\begin{aligned}\delta\lambda' &= - \sum_{\text{sources}} \int_0^T dt \sum_l \sum_k \sum_j \sum_m \epsilon_{lm}(\mathbf{x}, t) \delta_{jk} \delta_{lm} \frac{\partial \Psi_j}{\partial x_k}, \\ \delta\mu' &= - \sum_{\text{sources}} \int_0^T dt \sum_l \sum_k \sum_j \sum_m \epsilon_{lm}(\mathbf{x}, t) (\delta_{jl} \delta_{km} + \delta_{jm} \delta_{kl}) \frac{\partial \Psi_j}{\partial x_k}.\end{aligned}\tag{3.33}$$

and neglecting all wavefield components and derivatives in z-direction leads to

$$\begin{aligned}\delta\lambda' &= - \sum_{\text{sources}} \int_0^T dt \left(\epsilon_{xx} + \epsilon_{yy} \right) \left(\frac{\partial \Psi_x}{\partial x} + \frac{\partial \Psi_y}{\partial y} \right), \\ \delta\mu' &= - \sum_{\text{sources}} \int_0^T dt \left[\left(\epsilon_{xy} + \epsilon_{yx} \right) \left(\frac{\partial \Psi_x}{\partial y} + \frac{\partial \Psi_y}{\partial x} \right) \right] + 2 \left(\epsilon_{xx} \frac{\partial \Psi_x}{\partial x} + \epsilon_{yy} \frac{\partial \Psi_y}{\partial y} \right).\end{aligned}\tag{3.34}$$

Using the definition of the strain tensor ϵ_{ij} we get

$$\begin{aligned}\delta\lambda' &= - \sum_{\text{sources}} \int_0^T dt \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} \right) \left(\frac{\partial \Psi_x}{\partial x} + \frac{\partial \Psi_y}{\partial y} \right), \\ \delta\mu' &= - \sum_{\text{sources}} \int_0^T dt \left[\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) \left(\frac{\partial \Psi_x}{\partial y} + \frac{\partial \Psi_y}{\partial x} \right) \right] + 2 \left(\frac{\partial u_x}{\partial x} \frac{\partial \Psi_x}{\partial x} + \frac{\partial u_y}{\partial y} \frac{\partial \Psi_y}{\partial y} \right).\end{aligned}\tag{3.35}$$

Finally the gradients for the Lamé parameters λ , μ and the density ρ can be written as

$$\begin{aligned}\delta\lambda' &= - \sum_{\text{sources}} \int dt \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} \right) \left(\frac{\partial \Psi_x}{\partial x} + \frac{\partial \Psi_y}{\partial y} \right) \\ \delta\mu' &= - \sum_{\text{sources}} \int dt \left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \right) \left(\frac{\partial \Psi_x}{\partial y} + \frac{\partial \Psi_y}{\partial x} \right) + 2 \left(\frac{\partial u_x}{\partial x} \frac{\partial \Psi_x}{\partial x} + \frac{\partial u_y}{\partial y} \frac{\partial \Psi_y}{\partial y} \right) \\ \delta\rho' &= \sum_{\text{sources}} \int dt \left(\frac{\partial u_x}{\partial t} \frac{\partial \Psi_x}{\partial t} + \frac{\partial u_y}{\partial t} \frac{\partial \Psi_y}{\partial t} \right)\end{aligned}\tag{3.36}$$

3.4 Gradients for different model parametrizations

The gradients in terms of other material parameters $\mathbf{m}_{\text{new}}$ can be calculated by applying the chain rule on the Frechét kernel in the adjoint problem (Eq. (3.27)):

$$\delta \mathbf{m}_{\text{new}} = \sum_{\text{sources}} \int dt \sum_{\mathbf{R}} \left[\frac{\partial \mathbf{u}}{\partial \mathbf{m}} \frac{\partial \mathbf{m}}{\partial \mathbf{m}_{\text{new}}} \right]^* \delta \mathbf{u} \quad (3.37)$$

Using the relationships between P-wave velocity V_p , S-wave velocity V_s , the Lamé parameters λ , μ and density ρ :

$$V_p = \sqrt{\frac{\lambda + 2\mu}{\rho}}, \quad V_s = \sqrt{\frac{\mu}{\rho}} \quad (3.38)$$

or

$$\lambda = \rho V_p^2 - 2\rho V_s^2, \quad \mu = \rho V_s^2. \quad (3.39)$$

The gradient for V_p can be written as:

$$\begin{aligned} \delta V_p &= \sum_{\text{sources}} \int dt \sum_{\mathbf{R}} \left[\frac{\partial \mathbf{u}}{\partial \lambda} \frac{\partial \lambda}{\partial V_p} + \frac{\partial \mathbf{u}}{\partial \mu} \frac{\partial \mu}{\partial V_p} + \frac{\partial \mathbf{u}}{\partial \rho} \frac{\partial \rho}{\partial V_p} \right]^* \delta u_i \\ &= \sum_{\text{sources}} \int dt \sum_{\mathbf{R}} \left[\frac{\partial \mathbf{u}}{\partial \lambda} 2\rho V_p \right]^* \delta u_i \\ &= 2\rho V_p \sum_{\text{sources}} \int dt \sum_{\mathbf{R}} \left[\frac{\partial \mathbf{u}}{\partial \lambda} \right]^* \delta u_i \\ &= 2\rho V_p \delta \lambda \end{aligned} \quad (3.40)$$

The gradients for V_s and ρ are calculated in a similar way, so the gradients in terms of seismic velocities can be written as:

$$\begin{aligned} \delta V_p &= 2\rho V_p \delta \lambda, \\ \delta V_s &= -4\rho V_s \delta \lambda + 2\rho V_s \delta \mu, \\ \delta \rho_{\text{vel}} &= (V_p^2 - 2V_s^2) \delta \lambda + V_s^2 \delta \mu + \delta \rho \end{aligned} \quad (3.41)$$

3.5 Gradients for the stress-velocity code

The elastic forward problem and back-propagation of the data residuals is solved by using a time domain stress-velocity finite-difference (FD) code. Therefore the displacements in Eq. (3.36) have to be replaced by stresses and particle velocities $\mathbf{v}$ [e.g. Shipp and Singh, 2002]:

$$\begin{aligned} \delta \lambda' &= - \sum_{\text{sources}} \int dt \left[\frac{(\sigma_{xx} + \sigma_{yy})(\Sigma_{xx} + \Sigma_{yy})}{4(\lambda + \mu)^2} \right] \\ \delta \mu' &= - \sum_{\text{sources}} \int dt \left[\frac{\sigma_{xy} \Sigma_{xy}}{\mu^2} + \frac{1}{4} \left(\frac{(\sigma_{xx} + \sigma_{yy})(\Sigma_{xx} + \Sigma_{yy})}{(\lambda + \mu)^2} + \frac{(\sigma_{xx} - \sigma_{yy})(\Sigma_{xx} - \Sigma_{yy})}{\mu^2} \right) \right] \\ \delta \rho' &= - \sum_{\text{sources}} \int dt \left[\frac{\partial v_x}{\partial t} \Psi_x + \frac{\partial v_y}{\partial t} \Psi_y \right] \end{aligned} \quad (3.42)$$

were σ_{ij} and Σ_{ij} are the stresses of the forward and backpropagated wavefield, respectively. The displacements Ψ_i in the density gradient are calculated from the particle velocities by numerical integration.

3.6 Estimation of an optimum step length μ_n

The choice of the step length μ_n in Eq. 3.10 is crucial for the convergence of the steepest gradient optimization method. I demonstrate this using a very familiar test problem for optimization routines, the Rosenbrock function with two unknown parameters (Rosenbrock [1960], Fig. 3.4)

$$f_r(x, y) = (1 - x)^2 + 100(y - x^2)^2 \quad (3.43)$$

The aim is to find the minimum of this function located at the point [1,1] which is surrounded by a very narrow valley. We start the search for the minimum at [-0.5,0.5]. An obvious first choice would be a constant step length. Fig. 3.4 (top) shows the convergence after 16000 iteration steps of the steepest descent method when choosing a step length $\mu_n = 2e - 3$. Note the large model update during the first iteration step, when the gradient of the Rosenbrock function is large. After reaching the narrow valley the gradient is much smaller and as a result the model updates are also decreasing. This leads to a very slow convergence speed. Especially near the minimum the model updates become very small. When choosing a larger step length ($\mu_n = 2e - 3$, Fig. 3.4 (bottom)) the model update is larger even when the gradient is small, but the code fails to converge at all. Instead it is trapped in a narrow part of the valley. To solve this problem a variable step length is introduced. For three test step lengths μ_1 , μ_2 and μ_3 three test models are calculated

$$\begin{aligned} \mathbf{m}_{\text{test1}} &= \mathbf{m}_n + \mu_1 \delta \mathbf{m}'_n \\ \mathbf{m}_{\text{test2}} &= \mathbf{m}_n + \mu_2 \delta \mathbf{m}'_n \\ \mathbf{m}_{\text{test3}} &= \mathbf{m}_n + \mu_3 \delta \mathbf{m}'_n \end{aligned} \quad (3.44)$$

and the corresponding L2-norms $L2_1$, $L2_2$ and $L2_3$ are estimated (Fig. 3.5). The true misfit function (yellow line) can be approximated by fitting a parabola through the three points

$$L2_i = a\mu_i^2 + b\mu_i + c \quad (3.45)$$

where $i \in \{1, 2, 3\}$ and a , b , c are the unknown coefficients. This system of equations can be written as matrix equation:

$$\begin{pmatrix} \mu_1^2 & \mu_1 & 1 \\ \mu_2^2 & \mu_2 & 1 \\ \mu_3^2 & \mu_3 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} L2_1 \\ L2_2 \\ L2_3 \end{pmatrix}$$

or

$$\mathbf{Ax} = \mathbf{b}. \quad (3.46)$$

The unknown coefficients of this matrix equation are formally defined by

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}, \quad (3.47)$$

In the FWT code the solution vector $\mathbf{x}$ is calculated by Gaussian elimination. In the following the step length at the extremum of the parabola defines the optimum step length μ_{opt} (denoted as green square in Fig.3.5). This optimum step length is

$$\mu_{\text{opt}} = -\frac{b}{2a}. \quad (3.48)$$

The application of the variable step length calculation to the Rosenbrock test problem is shown in Fig. 3.6. The number of required iteration steps to reach the minimum is reduced by a factor 80 when compared with the constant step length gradient method. The only problem remaining is the slow convergence speed in the small valley of the Rosenbrock function, due to the fact that the update occurs along the gradient direction of the objective function resulting in a "criss-cross" pattern. This behaviour can be avoided by applying a nonlinear conjugate gradient method (chapter 3.7). In case of the FWT algorithm the three test step lengths for the individual material parameter classes are calculated by scaling the maximum of the gradient to the maximum of the actual models:

$$\begin{aligned} \mu_\lambda &= p \frac{\max(\lambda_n)}{\max(\delta \lambda_n)} \\ \mu_\mu &= p \frac{\max(\mu_n)}{\max(\delta \mu_n)} \\ \mu_\rho &= p \frac{\max(\rho_n)}{\max(\delta \rho_n)} s_{\text{rho}} \end{aligned} \quad (3.49)$$

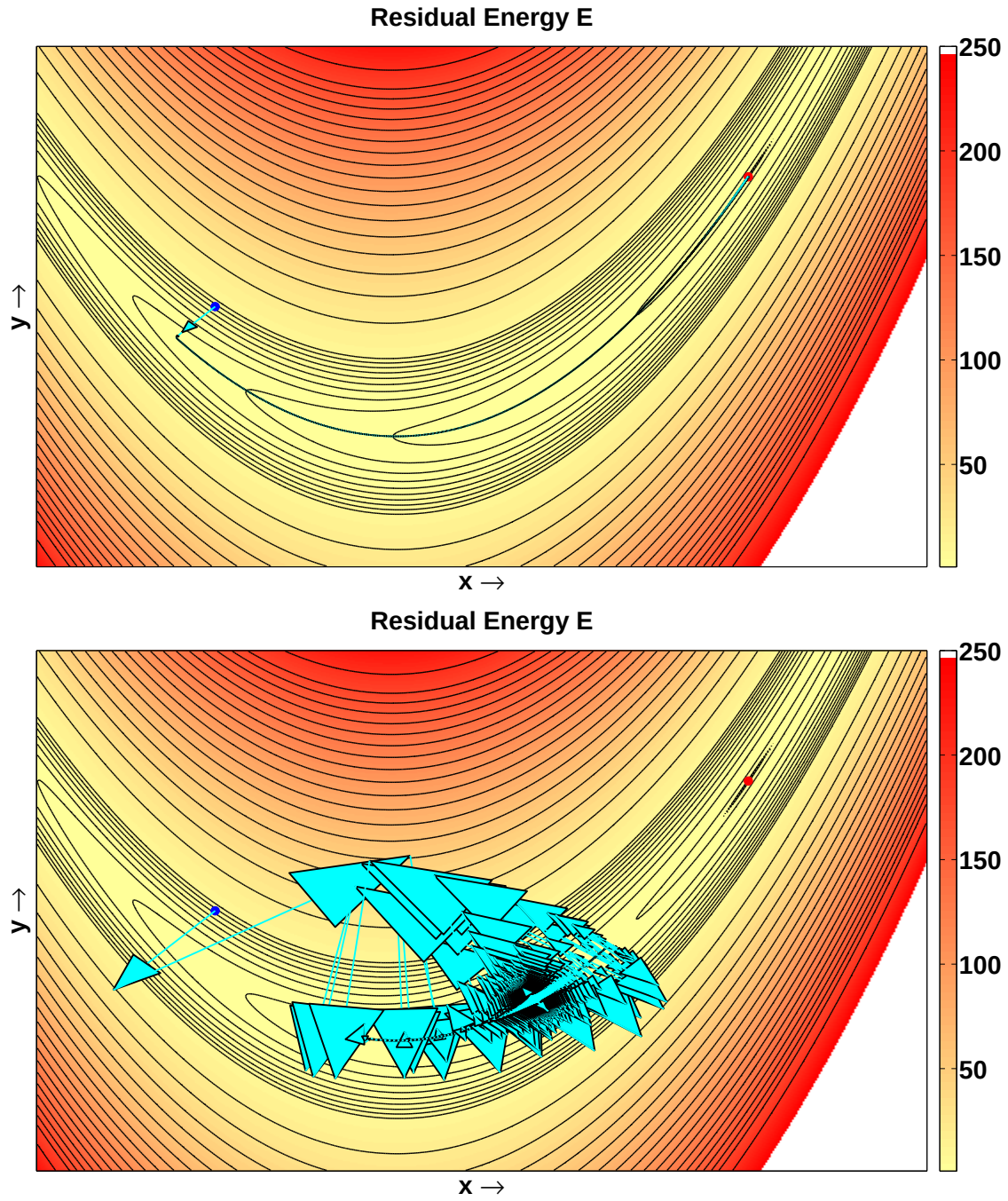


Figure 3.4: Results of the convergence test for the Rosenbrock function. The minimum is marked with a red point, the starting position with a blue point. The maximum number of iterations is 16000. The step length μ_n varies between $2e - 3$ (top) and $6.1e - 3$ (bottom).

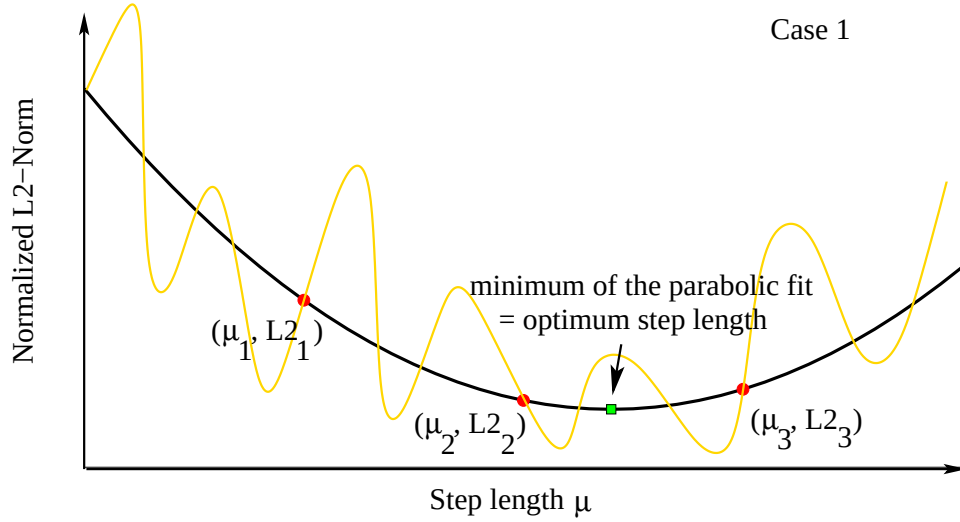


Figure 3.5: Line search algorithm to find the optimum step length μ_{opt} : The true misfit function (yellow line) is approximated by a parabola fitting values of the objective function for 3 different step length.

Because changes of the density model are in most cases smaller than velocity changes the step length for the density update can be systematically reduced by a factor s_{rho} . All material parameters can be updated simultaneously or according to a hierarchical strategy. To save computational time the corresponding L_2 -norms are calculated for a few representative shots. The number of shots depends on the complexity of the problem and the signal/noise ratio of the data. For the acoustic case the step length estimation by parabolic fitting works very well and leads to a smooth decrease of the misfit function during the FWT (Kurzmann (2007), personal communication, ?). For the multiparameter elastic FWT the misfit function consists of more local minima and therefore the decrease of the objective function is not as smooth as in the acoustic case. Brossier [2009] proposed a more intensive bracketing stage before applying the parabolic fit. Starting from $p_1 = 0.0$ the test step lengths p_2 and p_3 are calculated to satisfy the following criteria:

$$\begin{aligned} L2_2(\mathbf{m}_{\text{test}2} = \mathbf{m}_n + \mu_2 \delta \mathbf{m}'_n) &< L2_1(\mathbf{m}_{\text{test}1} = \mathbf{m}_n) \\ L2_3(\mathbf{m}_{\text{test}3} = \mathbf{m}_n + \mu_3 \delta \mathbf{m}'_n) &> L2_2(\mathbf{m}_{\text{test}2} = \mathbf{m}_n + \mu_2 \delta \mathbf{m}'_n) \end{aligned} \quad (3.50)$$

This approach leads to a smoother decrease of the objective function, but also increases the number of required forward models.

3.7 Nonlinear Conjugate Gradient Method

To increase the convergence speed in narrow valleys it would be better to update the model at iteration step n not exactly along the gradient direction $\delta \mathbf{m}_n$, but along the conjugate direction $\delta \mathbf{c}_n$

$$\delta \mathbf{c}_n = \delta \mathbf{m}_n + \beta_n \delta \mathbf{c}_{n-1}, \quad (3.51)$$

The first iteration step ($n=1$) consists of a model update along the steepest descent direction:

$$\mathbf{m}_2 = \mathbf{m}_1 + \mu_1 \delta \mathbf{m}_1, \quad (3.52)$$

For all subsequent iteration steps ($n > 1$) the model is updated along the conjugate direction:

$$\mathbf{m}_{n+1} = \mathbf{m}_n + \mu_n \delta \mathbf{c}_n, \quad (3.53)$$

where $\delta \mathbf{c}_1 = \delta \mathbf{m}_1$. The weighting factor β can be calculated in different ways [Nocedal and Wright, 2006]:

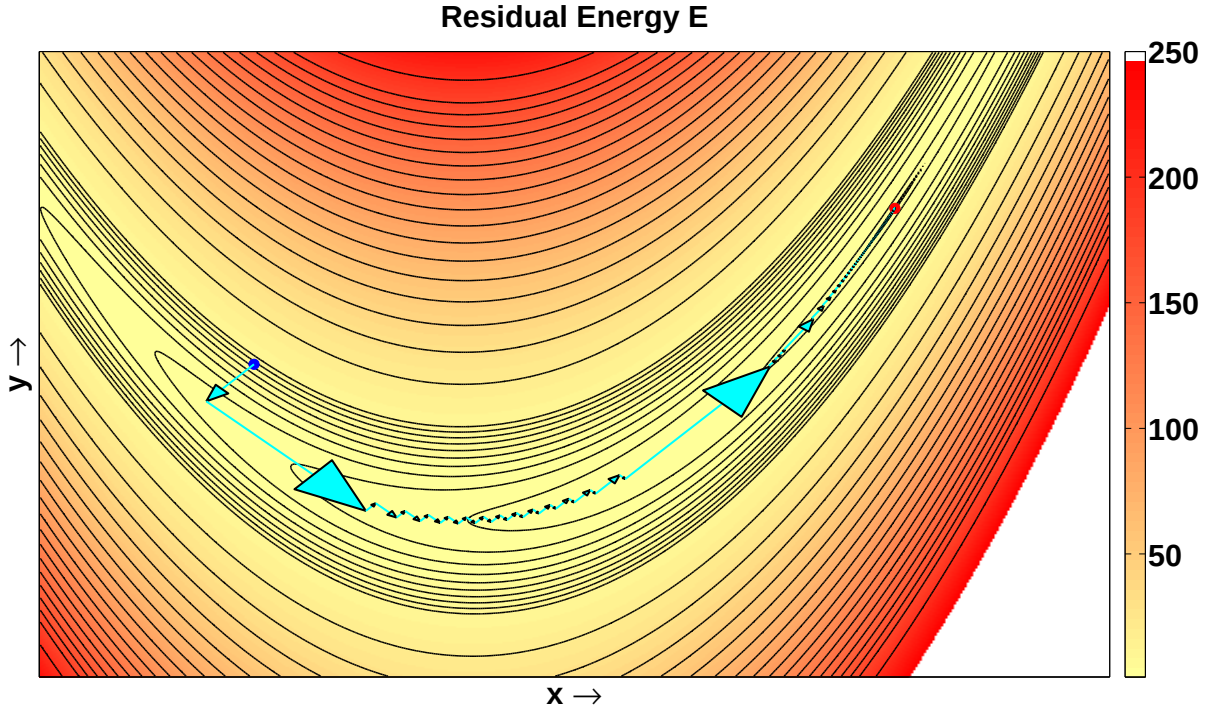


Figure 3.6: Results of the convergence test for the Rosenbrock function using the pure gradient method. The minimum is marked by a red point, the starting position by a blue point. The number of iterations is 200. The optimum step length is calculated at each iteration by the parabola fitting algorithm. Note the criss-cross pattern of the updates in the narrow valley and the slow convergence speed near the minimum.

1. Fletcher-Reeves [Fletcher and Reeves, 1964]:

$$\beta_n^{\text{FR}} = \frac{\delta \mathbf{m}_n^T \delta \mathbf{m}_n}{\delta \mathbf{m}_{n-1}^T \delta \mathbf{m}_{n-1}} \quad (3.54)$$

2. Polak-Ribière [Polak and Ribière, 1969]:

$$\beta_n^{\text{PR}} = \frac{\delta \mathbf{m}_n^T (\delta \mathbf{m}_n - \delta \mathbf{m}_{n-1})}{\delta \mathbf{m}_{n-1}^T \delta \mathbf{m}_{n-1}} \quad (3.55)$$

3. Hestenes-Stiefel [Hestenes and Stiefel, 1952]:

$$\beta_n^{\text{HS}} = \frac{\delta \mathbf{m}_n^T (\delta \mathbf{m}_n - \delta \mathbf{m}_{n-1})}{\delta \mathbf{c}_{n-1}^T (\delta \mathbf{m}_n - \delta \mathbf{m}_{n-1})} \quad (3.56)$$

I use the very popular choice $\beta_n = \max[0, \beta_n^{\text{PR}}]$ which provides an automatic direction reset. This is important because subsequent search directions lose conjugacy requiring the search direction to be reset to the steepest descent direction. Note that the conjugate gradient method doesn't require any additional computational time because only the gradient $\delta \mathbf{m}_n$ at two subsequent iterations has to be known. The application of the nonlinear conjugate gradient method combined with the variable step length calculation to the Rosenbrock function is shown in Fig. 3.7. The criss-cross pattern of the steepest descent method has vanished and the conjugate gradient method already converges after 30 iterations compared with 200 iteration steps of the pure gradient method.

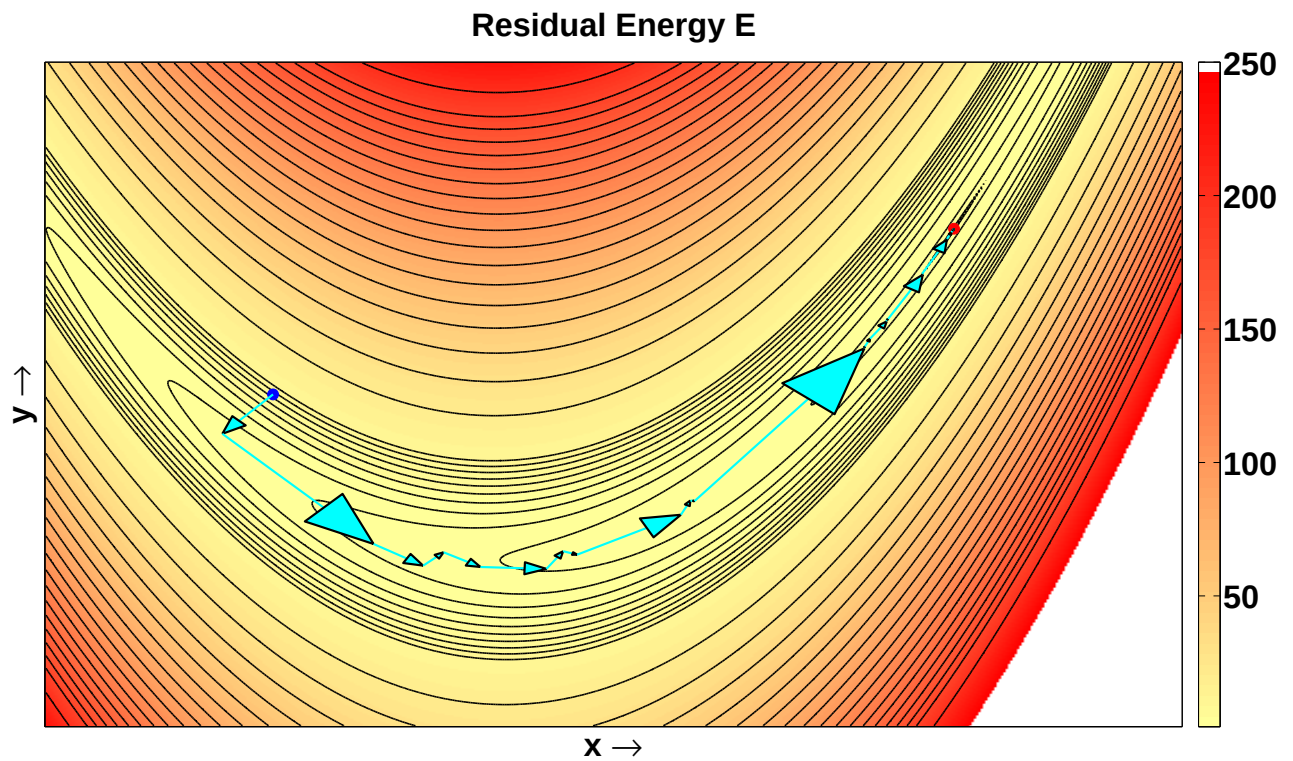


Figure 3.7: Results of the convergence test for the Rosenbrock function using the nonlinear conjugate gradient method, where the optimum step length is calculated with the parabolic fitting algorithm. The minimum is marked by a red cross, the starting point by a blue point. The number of iterations is 30.

3.8 The elastic FWT algorithm

In summary the FWT algorithm consists of the following steps:

1. Define a starting model $\mathbf{m}_1$ in the parameter space. This model should represent the long wavelength part of the underground very well, because the FWT code is only capable to reconstruct structures at or below the dominant seismic wavelength due to its slow convergence speed, the nonlinearity of the problem and the inherent use of the Born approximation to calculate the gradient direction.
2. At iteration step n do:
 - (a) For each shot solve the forward problem, stated in Eq.(3.16) for the actual model $\mathbf{m}_n$ to generate a synthetic dataset $\mathbf{u}^{\text{mod}}$ and the wavefield $\mathbf{u}(\mathbf{x}, t)$.
 - (b) Calculate the residual seismograms $\delta\mathbf{u} = \mathbf{u}^{\text{mod}} - \mathbf{u}^{\text{obs}}$ for the x- and y-components of the seismic data.
 - (c) Generate the wavefield $\Psi(\mathbf{x}, t)$ by backpropagating the residuals from the receiver positions.
 - (d) Calculate the gradients $\delta\mathbf{m}_n$ of each material parameter according to Eqs.(3.36).
 - (e) To increase the convergence speed an appropriate preconditioning operator P is applied to the gradient $\delta\mathbf{m}$

$$\delta\mathbf{m}_n^p = P\delta\mathbf{m}_n \quad (3.57)$$

Examples of simple preconditioning operator are given in chapter ?? for a cross-well acquisition geometry and in chapter 7.1.3 for a reflection geometry.

- (f) For a further increase of the convergence speed calculate the conjugate gradient direction for iteration steps $n \geq 2$:

$$\delta\mathbf{c}_n = \delta\mathbf{m}_n^p + \beta\delta\mathbf{c}_{n-1}, \text{ with } \delta\mathbf{c}_1 = \delta\mathbf{m}_1^p \quad (3.58)$$

where the weighting factor

$$\beta^{\text{PR}} = \delta\mathbf{m}_n^p \frac{\delta\mathbf{m}_n^p - \delta\mathbf{m}_{n-1}^p}{\delta\mathbf{m}_{n-1}^p \delta\mathbf{m}_{n-1}^p} \quad (3.59)$$

by Polak-Ribière is used. The convergence of the Polak-Ribière method is guaranteed by choosing $\beta = \max[\beta^{\text{PR}}, 0]$.

- (g) Estimate the step length μ_n by the line search algorithm described in chapter 3.6.
- (h) Update the material parameters using the gradient method

$$\mathbf{m}_{n+1} = \mathbf{m}_n - \mu_n \delta\mathbf{c}_n. \quad (3.60)$$

If the material parameters are not coupled by empirical relationships it is important to update all three elastic material parameters at the same time, otherwise strong artifacts may dominate the inversion result, especially in the case of very complex media.

3. If the residual energy E is smaller than a given value stop the iteration. Otherwise continue with the next iteration step.

Chapter 4

Source Wavelet Inversion

So far we assumed that only the material parameters are unknown, while the characteristics of the sources are perfectly known. For the application of FWI to a field dataset the source wavelet has to be estimated. In frequency domain the source wavelet s has the complex value

$$s = e + if$$

where $i = \sqrt{-1}$, e and f are the real and imaginary parts, respectively. The seismograms of the vertical displacements of the modelled data can be described by:

$$v_r^m = (c_{v,r} + id_{v,r})(e + if)$$

where $(c_{v,r} + id_{v,r})$ denotes the spike response of the vertical displacement and r the receiver location. Similar the seismograms of the vertical displacements of the field data are:

$$v_r^d = (a_{v,r} + ib_{v,r})$$

Update the real and imaginary parts of the source wavelet with the Newton method

$$\begin{aligned} \mathbf{e}_{n+1} &= \mathbf{e}_n - \mathbf{H}_n^{-1} \left(\frac{\partial E}{\partial e} \right)_n, \\ \mathbf{f}_{n+1} &= \mathbf{f}_n - \mathbf{H}_n^{-1} \left(\frac{\partial E}{\partial f} \right)_n. \end{aligned} \quad (4.1)$$

where $\mathbf{H}$, $\left(\frac{\partial E}{\partial e} \right)_n$ and $\left(\frac{\partial E}{\partial f} \right)_n$ are the Hessian matrix and gradient vector for e and f , respectively. With the objective function

$$E = \frac{1}{2} \sum_r (v_r^m - v_r^d)(v_r^m - v_r^d)^* \quad (4.2)$$

the gradients and Hessian can be explicitly calculated

$$\begin{aligned} \frac{\partial E}{\partial e} &= \sum_r [e(c_{v,r}^2 + d_{v,r}^2) - a_{v,r}c_{v,r} - b_{v,r}d_{v,r}] \\ \frac{\partial E}{\partial f} &= \sum_r [f(c_{v,r}^2 + d_{v,r}^2) + a_{v,r}d_{v,r} - b_{v,r}c_{v,r}] \\ \frac{\partial^2 E}{\partial e^2} &= \sum_r (c_{v,r}^2 + d_{v,r}^2), \\ \frac{\partial^2 E}{\partial f \partial e} &= 0, \quad \frac{\partial^2 E}{\partial e \partial f} = 0, \\ \frac{\partial^2 E}{\partial f^2} &= \sum_r (c_{v,r}^2 + d_{v,r}^2). \end{aligned} \quad (4.3)$$

Inserting Eq. (4.3) in Eq. (4.1) leads to

$$\begin{aligned} \mathbf{e}_{n+1} &= \frac{\sum_r (a_{v,r} c_{v,r} - b_{v,r} d_{v,r})}{\sum_r (c_{v,r}^2 + d_{v,r}^2)} \\ \mathbf{f}_{n+1} &= -\frac{\sum_r (a_{v,r} d_{v,r} - b_{v,r} c_{v,r})}{\sum_r (c_{v,r}^2 + d_{v,r}^2)}. \end{aligned} \quad (4.4)$$

To stabilize the inversion a Marquardt-Levenberg regularization is required:

$$\begin{aligned} \mathbf{e}_{n+1} &= \mathbf{e}_n - (\mathbf{H}_n + \lambda_e \mathbf{I})^{-1} \left(\frac{\partial E}{\partial \mathbf{e}} \right)_n \\ \mathbf{f}_{n+1} &= \mathbf{f}_n - (\mathbf{H}_n + \lambda_f \mathbf{I})^{-1} \left(\frac{\partial E}{\partial \mathbf{f}} \right)_n \end{aligned}$$

where λ_e, λ_f are damping factors and $\mathbf{I}$ the unity matrix. Therefore Eq. (4.4) can be written as

$$\begin{aligned} \mathbf{e}_{n+1} &= \frac{\sum_r (a_{v,r} c_{v,r} - b_{v,r} d_{v,r})}{\sum_r (c_{v,r}^2 + d_{v,r}^2) + \lambda_e} \\ \mathbf{f}_{n+1} &= -\frac{\sum_r (a_{v,r} d_{v,r} - b_{v,r} c_{v,r})}{\sum_r (c_{v,r}^2 + d_{v,r}^2) + \lambda_f}. \end{aligned} \quad (4.5)$$

The values of the damping factors can be expressed as fractions of the maximum value in the denominators

$$\lambda_e = \lambda_f = \epsilon_{stf} \max_r \left| \sum_r (c_{v,r}^2 + d_{v,r}^2) \right| \quad (4.6)$$

This approach is a stabilized Wiener deconvolution.

Chapter 5

Getting Started

In the following sections, we give a short description of the different modeling parameters, options and how the program is used in a parallel MPI environment.

5.1 Requirements

The parallelization employs functions of the Message Passing Interface (MPI). MPI has to be installed when compiling and running the DENISE software. At least two implementations exist for Unix-based networks: OpenMPI, MPICH2 and Intel-MPI. The LAM-MPI implementation is no longer supported by the developers. However, currently all four implementations work with DENISE. OpenMPI, MPICH2 and Intel-MPI are MPI programming environments and development systems for heterogeneous computers on a network. As of the time of writing we get the best performance out of DENISE by using Intel-MPI together with the latest Intel-Compiler on a NEC-Linux Cluster. With MPI a dedicated cluster or an existing network computing infrastructure can act as a parallel computer. Fast network (infiniband) connections and RAM access are the most important issues for a good scaling of the DENISE code. The latest version of OpenMPI can be obtained from <http://www.open-mpi.org>. MPICH2 is available at <http://www-unix.mcs.anl.gov/mpi/mpich>. LAM-MPI can be downloaded here: <http://www.lam-mpi.org>, the commercial Intel-MPI from here: <https://software.intel.com/en-us/intel-mpi-library>.

5.1.1 LAM

Even though outdated, LAM-MPI will be used to illustrate the setting up of the MPI implementation. In order to reproduce the following instructions, you must first install the LAM-MPI software. On SUSE LINUX systems the package can be installed with `yast`. The software can also be downloaded from <http://www.lam-mpi.org>. A good documentation of LAM/MPI is available at this site. Before you can run the DENISE software on more than one node you must boot LAM. This requires that you can connect to all of your nodes with `ssh` without a password. To enable `ssh` connection without a password you must generate authentication keys on each node using the command `ssh-keygen -t rsa`. These are saved in the file `~/.ssh/id_rsa.pub`. Copy all authentication keys into one file named `authorized_keys` and copy this file on all nodes into the directory `~/.ssh`.

Before you can boot LAM you must specify the IP addresses of the different processing elements in an ASCII file. An example file is `par/lamhosts`. Each line must contain one IP address. The first IP number corresponds to node 0, the second line to node 1 and so forth. Note that in older LAM-MPI implementations the `mpirun` command to run the FD programs must always be executed on node 0 !, i.e. you must log on node 0 (first line in the file `par/lamhosts`) and run the software on this node. In the last stable release of LAM-MPI, the node 0 just has to be listed in the `lamhosts` file (the order does not matter). To boot lam just do `lamboot -v par/lamhosts`. To run the software in serial on a single PC just type `lamboot` without specifying a `lamhosts` file. You can still specify more PEs in the FD software but all processes will be executed on the same CPU. Thus this only makes sense if you run the software on a multicore system. In this case, you should boot it without a `lamhosts` file and specify a total number of 2 processing elements (PEs). To shut down LAM again (before logout) use the command `lamclean -v`. However, it is not necessary to shut down/restart LAM after each logout/login.

5.1.2 How to run DENISE on the NEC-Linuxcluster at RZ Kiel

Before you can run DENISE on the Linux cluster at the computing centre in Kiel you have initialize Intel-MPI and Intel-compilers, and assure that the different nodes can communicate password-free. This has to be done only once.

1. Add the following lines to your `.bashrc` in your `$HOME` directory, to initialize Intel-MPI and the Intel-compiler:

```
. /opt/intel/composer_xe_2013_sp1/bin/compilervars.sh intel64
. /opt/intel/impi/4.1.1.036/intel64/bin/mpivars.sh
```

2. To setup a password-free communication between the different nodes generate a pair of authentication keys for ssh with:

```
[sungwXXX@nesh-fe2 ~]$ ssh-keygen -t dsa
```

You can accept the default values by hitting <return>.

3. Copy the file `$HOME/.ssh/id_dsa.pub` to `$HOME/.ssh/authorized_keys`.

Because DENISE can produce up to a few GB of data output, don't run the code from the home-directory. To submit a batch job it is required, that DENISE is located in the `$WORK` directory. Keep in mind, that the file system `$WORK` will not be automatically backedup, so do a manual backup from time to time. After compiling the code (see section 5.3), you can define and start a batch job with a shell script like this:

```
#!/bin/ksh
#PBS -l elapstim_req=48:00:00 # walltime
#PBS -l cputim_job=768:00:00 # akkumulated CPU-time per node
#PBS -l memsz_job=100gb      # RAM requirement
#PBS -b 1                   # number of nodes
#PBS -T intmpi              # job type; intmpi for Intel-MPI
#PBS -l cpunum_job=16       # number of cores per node
#PBS -N DENISE              # name of the batch job
#PBS -o DENISE.out          # file for standard output
#PBS -j o                   # standard and error output
#PBS -q clmedium            # batch class
#
#
. /opt/intel/composer_xe_2013_sp1/bin/compilervars.sh intel64
. /opt/intel/impi/4.1.1.036/intel64/bin/mpivars.sh
. /opt/modules/Modules/3.2.6/init/ksh
#
#
cd $WORK/DENISE_PSV/par
mpirun $NQSII_MPIOPTS -np 15 ../bin/denise DENISE_marm_OBC.inp FWI_workflow_marmousi.inp
```

The individual parameters and possible batch-job classes are described in more detail on the homepage of the RZ Kiel https://www.rz.uni-kiel.de/hpc/nec_cluster.html The most important parameters are

- *elapstim_req*, which defines how long the job will actually run
- *cputim_job* the accumulated CPU-time per node
- *memsz_job* the required memory per node
- *-b* the number of nodes
- *mpich* the job type, in case of Intel-MPI you have to choose *mpich*
- *cpunum_job* number of CPUs per node

- *-N* name of the batch job
- *-o* file name for standard output
- *-q* the requested batch-class.

An example for a job-file can be found in the /DENISE/jobs directory. The job can be submitted with

```
[sungwXXX@rzcluster ~]$ qsub DENISE.job
```

With

```
[sungwXXX@rzcluster ~]$ qstat
```

you can check the status of your Jobs and with

```
[sungwXXX@rzcluster ~]$ qdel <job_id>
```

you can cancel a submitted or running job, where *<job_id>* denotes the number at the first column of the status information, f.e.

```
[sungwXXX@nesh-fe2 jobs]$ qstat
```

RequestID	ReqName	UserName	Queue	Pri	STT	S	Memory	CPU	Elapse	R	H	M	Jobs
459654.ace-ssio	DENISE	sungwXXX	clmedium	0	RUN	-	13.89G	1077746.75	68303	Y	Y	Y	Y
470371.ace-ssio	SAVA	sungwXXX	clmedium	0	QUE	-	0.00B	0.00	0	Y	Y	Y	Y

```
[sungwXXX@nesh-fe2 jobs]$ qdel 459654
```

would kill the first job in the queue. For further information I again refer to the homepage of the RZ-Kiel: https://www.rz.uni-kiel.de/hpc/nec_cluster.html

5.2 Installation

DENISE consists of four different packages:

- The source code
- A collection of benchmark models and pre-/postprocessing tools
- Matlab scripts for data pre-processing.
- The manuals for the different code versions.

Start with unpacking the source code package (e.g. by *tar -zxvf DENISE.tgz*) and changing to the directory DENISE (*cd DENISE*) you will find different subdirectories:

.svn

The software is updated using the version control system subversion (SVN). This directory contains internal information on recent software updates etc. You can ignore this directory, a clean release should not have any .svn directories at all. Each subdirectory described below also contains a subdirectory SVN which can be ignored as well.

bin

This directory contains all executable programs, generally DENISE and snapmerge. These executables are generated using the command *make <program>* (see below).

jobs

This directory contains Batch-scripts to submit SAVA modelling/inversion runs on HPCs with PBS-batch system.

libcseife

This directory contains external contributions to DENISE for the implementation of a Butterworth frequency filter.

mfiles

Here some Matlab routines (m-files) are stored. These Matlab programs can be used to find optimal relaxation frequencies to approximate a constant Q (qapprox.m) or to plot Q as a function of frequency for certain relaxation frequencies

and value of tau (qplot.m). For further details on the theory behind these algorithms we refer to the Phd thesis of T. Bohlen Bohlen [1998] and to the paper in which the so-called tau-method is described Blanch et al. [1995]. The parameter tau is used in this software to define the level of attenuation.

par

Parameter files for DENISE modelling and inversion.

src

This directory contains the complete source codes.

5.3 Compilation of DENISE

Before compiling DENISE you have to compile the additional library for timedomain filtering. In the DENISE/libcseife directory simply use the shell script:

```
-bash-2.05b$ :~/DENISE/libcseife> make
```

It is necessary to preinstall FFTW - Fastest Fourier Transform in the West (<http://www.fftw.org/>).

The source code of DENISE is located in the directory DENISE/src. To compile DENISE the name of the model function has to be entered in the MAKEFILE. Depending on your MPI environment (MPI distribution) you may need to modify the compiler options in src/Makefile. For a few typical platforms the compiler options are available in src/Makefile. It is often useful to enable a moderate level of optimization (typically -O3). The highest level of optimization -O4 can lead to a strong performance improvement. For example the optimization option -O4 of the hcc LAM compiler leads to a speedup of DENISE of approximately 30 per cent. Even though, keep in mind that -O4 can also lead to crashes and compilation errors, when used in combination with certain compilers. Linbin Zhang suggested to use Advanced Vector Extensions (AVX) on supported CPUs by using the following compiler flags:

```
LFLAGS=-lm -lcseife -lfftw3 -ftz -lstdc++
CFLAGS=-O3 -xAVX -ipo -fno-fnalias -restrict
```

Depending e.g. on the size of the inversion problem, used spatial FD-operators, a reduction of computation time of up to 20 % could be measured. No other changes in the Makefile should be necessary.

```
# Makefile for DENISE

#-----
# edit here:

# source code for model generation

MODEL_EL = half_space.c
EXEC= ../bin

# Compiler (LAM: CC=hcc, CRAY T3E: CC=cc)

# ON Linux cluster running LAM (options for DENISE)
#CC=hcc
#LFLAGS=-lm -lmpi -lcseife
#CFLAGS=-O3
#SFLAGS=-L../libcseife
#IFLAGS=-I../libcseife

# On CRAY T3E
# CC=cc
```

```

# On NEC-cluster with Intel-MPI
CC=mpiicc
LFLAGS=-lm -lcseife -lfftw3 -ftz -lstdc++
CFLAGS=-O3 -xAVX -ipo -fno-fnalias -restrict
SFLAGS=-L./../libcseife -L$//sfs/fs4/work-sh2/sungw331/fftw-3.3.4/lib
IFLAGS=-I./../libcseife -I$//sfs/fs4/work-sh2/sungw331/fftw-3.3.4/include

# On MARWIN
# CC=mpiicc
# LFLAGS=-lm -lcseife -lfftw3 -lstdc++
# CFLAGS=-O3
# SFLAGS=-L./../libcseife
# IFLAGS=-I./../libcseife

# On Desktop computer with LinuxMint 17, OpenMPI and gcc 4.8.2
#CC=mpicc
#LFLAGS=-lm -lcseife -lfftw3 -lstdc++
#CFLAGS=-O3 -fno-stack-protector
#SFLAGS=-L./../libcseife
#IFLAGS=-I./../libcseife

# On HLRN system
#CC=mpcc
#LFLAGS=-lm

# ALTIX
#CC=icc
#CFLAGS=-mp -O3 -ipo
#LFLAGS=-lmpi -lm -i-static

# after this line, no further editing should be necessary
# -----

```

To compile the program DENISE you must change to the src directory and execute:

```
-bash-2.05b$ :~/DENISE/src> make denise
```

The following (or a similar) output should occur:

```

[...]
mpicc -O3 -c sources.c -I./../libcseife
mpicc -O3 -c solvelin.c -I./../libcseife
mpicc -O3 -c spat_filt.c -I./../libcseife
mpicc -O3 -c splitsrc.c -I./../libcseife
mpicc -O3 -c splitsrc_back.c -I./../libcseife
mpicc -O3 -c splitrec.c -I./../libcseife
mpicc -O3 -c stalta.c -I./../libcseife
mpicc -O3 -c step_length_est.c -I./../libcseife
mpicc -O3 -c step_length_est1.c -I./../libcseife
mpicc -O3 -c stf.c -I./../libcseife
mpicc -O3 -c taper.c -I./../libcseife
mpicc -O3 -c taper_grad.c -I./../libcseife
mpicc -O3 -c taper_grad_shot.c -I./../libcseife
mpicc -O3 -c timedomain_filt.c -I./../libcseife
mpicc -O3 -c time_window.c -I./../libcseife
mpicc -O3 -c util.c -I./../libcseife

```

```
mpicc -O3 -c wavelet.c -I./../libcseife
mpicc -O3 -c wavelet_stf.c -I./../libcseife
mpicc -O3 -c writemod.c -I./../libcseife
mpicc -O3 -c write_par.c -I./../libcseife
mpicc -O3 -c writedsk.c -I./../libcseife
mpicc -O3 -c zero_fdveps.c -I./../libcseife
mpicc -L./../libcseife denise.o calc_mat_change.o calc_mat_change_test.o
calc_res.o calc_opt_step.o calc_opt_step_test.o calc_energy.o catseis.o
checkfd_ssg_elastic.o conv_FD.o psource.o holbergcoeff.o comm_ini.o
exchange_v.o exchange_s.o exchange_L2.o fft.o fft_filt.o forward_mod.o
[...] snap_ssg.o seismo_ssg.o surface_elastic_2nd.o writemod.o
write_par.o writedsk.o zero_fdveps.o -o ../bin/denise -lm -lcseife
```

The program `snapmerge` that is used to merge the snapshots (see below) can be compiled with "make `snapmerge`". Since this is not a MPI program (no MPI functions are called) the MPI libraries are not required and any standard compiler (like `gcc` and `cc`) can be used to compile this program. The executables `denise` and `snapmerge` are located the directory `bin`. You can also compile `DENISE` and `snapmerge` from the `/par` directory using the shell script `compile-DENISE.sh`.

5.4 Running the program

Each `DENISE` run reads the required parameters from the parameter files `par/DENISE.inp` and `par/FWI_workflow.inp`. A detailed description of the parameters can be found in chapter 6. The command to start a simulation on 8 processor with the lowest priority of -19 (in order to allow working on the a workstation while running a simulation) is as follows. Please note, that we assume you have navigated to the folder `DENISE/par` and all parameter files are located in this directory.

```
mpirun -np 8 nice -19 ../bin/denise DENISE.inp FWI_workflow.inp
```

If you use LAMMPI the command `lamboot -v lamhost` must be executed on node 0 which is the PE where `./par/lamhosts` is the file containing IP addresses of all computing nodes. It is often useful to save the standard output of the program for later reference. The screen output may be saved to `DENISE.out` using

```
mpirun -np 8 nice -19 ../bin/denise DENISE.inp FWI_workflow.inp > DENISE.out
```

After the output of geometry and model parameters the code starts the time stepping and displaying timing information:

```
=====
MYID=0 ***** Starting simulation (forward model) for shot 1 of 1 *****
=====

Number of samples (nts) in source file: 3462
Number of samples (nts) in source file: 3462
Message from function wavelet written by PE 0
1 source positions located in subdomain of PE 0
have been assigned with a source signal.

PE 0 outputs source time function in SU format to start/source_signal.0.su.shot1

Computing timestep 1000 of 3462

**Message from update_v (printed by PE 0):
Updating particle velocities ... finished (real time: 0.00 s).
```

```

particle velocity exchange between PEs ... finished (real time: 0.00 s).

**Message from update_s (printed by PE 0):
Updating stress components ... finished (real time: 0.00 s).
stress exchange between PEs ... finished (real time: 0.00 s).
total real time for timestep 1000 : 0.01 s.

Computing timestep 2000 of 3462

**Message from update_v (printed by PE 0):
Updating particle velocities ... finished (real time: 0.00 s).
particle velocity exchange between PEs ... finished (real time: 0.00 s).

**Message from update_s (printed by PE 0):
Updating stress components ... finished (real time: 0.00 s).
stress exchange between PEs ... finished (real time: 0.00 s).
total real time for timestep 2000 : 0.01 s.

Computing timestep 3000 of 3462

**Message from update_v (printed by PE 0):
Updating particle velocities ... finished (real time: 0.00 s).
particle velocity exchange between PEs ... finished (real time: 0.00 s).

**Message from update_s (printed by PE 0):
Updating stress components ... finished (real time: 0.00 s).
stress exchange between PEs ... finished (real time: 0.00 s).
total real time for timestep 3000 : 0.01 s.
PE 0 is writing 11 seismograms (vx) to
    su/DENISE_US_x.su.shot1.it1
PE 0 is writing 11 seismograms (vy) to
    su/DENISE_US_y.su.shot1.it1

**Info from main (written by PE 0):
CPU time of program per PE: 17 seconds.
Total real time of program: 18.08 seconds.
Average times for
velocity update:      0.003 seconds
stress update:        0.002 seconds
velocity exchange:    0.000 seconds
stress exchange:      0.000 seconds
timestep:             0.005 seconds

```

5.5 Postprocessing

The wavefield snapshots can be merged using the program *snapmerge*. The program *snapmerge* is not a MPI program. Therefore, it can be executed without MPI and the *mpirun* command. You can run *snapmerge* on any PC since a MPI environment (e.g. LAM) is not required. You may therefore copy the snapshot outputs of the different nodes to another non-MPI computer to merge the files together. *snapmerge* reads the required information from the DENISE parameter file. Simply type

```
-bash-2.05b$~/DENISE/par> ../bin/snapmerge DENISE.inp
```

Depending on the model size the merge process may take a few seconds or hours. The output should read like this:

```
pressure (files: ./snap/test.bin.p.??? ).  
  
writing merged snapshot file to  ./snap/test.bin.p  
Opening snapshot files: ./snap/test.bin.p.???  ... finished.  
Copying... ... finished.  
Use  
xmovie n1=100 n2=100 < ./snap/test.bin.p loop=1 label1=Y label2=X title=%g  
to play movie.
```

Chapter 6

Definition of parameters for the modeling and inversion code

The geometry of the FD grid and all parameters for the wavefield simulation and inversion have to be defined in a parameter file (which we name in this case *DENISE/par/DENISE.inp*). Parameters changing during the waveform inversion are defined in a separate file (which we name in this case *DENISE/par/FWI_workflow.inp*). This allows the flexible combination of different inversion parameters and therefore an implementation of complex FWI workflows. In the following we will explain every input parameter section in detail.

6.1 Input file with fixed parameters DENISE.inp

Most lines in the parameter file are organized as follows:

```
description_of_parameter_(VARNAME)_(switches) = parameter value
```

where VARNAME denotes the name of the global variable in which the value is saved in all functions of the program. The possible values are described in switches. A comment line is indicated by a # on the very first position of a line. The meaning of the different parameters is described in the following.

Domain decomposition

```
#----- Domain Decomposition -----  
number_of_processors_in_x-direction_(NPROCX) = 2  
number_of_processors_in_y-direction_(NPROCY) = 2  
#
```

Parallelization is based on domain decomposition (see Figure 6.1), i.e each processing element (PE) updates the wavefield within his portion of the grid. The model is decomposed by the program into sub grids. After decomposition each processing elements (PE) saves only his sub-volume of the grid. NPROCX and NPROCY specify the number of processors in x-, y-direction, respectively (Figure 6.1). The total number of processors thus is $NP=NPROCX*NPROCY$. This value must be specified when starting the program with the mpirun command: *mpirun -np <NP> ../bin/DENISE DENISE.inp* (see section 5.4). If the total number of processors in DENISE.inp and the command line differ, the program will terminate immediately with a corresponding error message. Obviously, the total number of PEs ($NPROCX*NPROCY$) used to decompose the model should be less equal than the total number of CPUs which are available on your parallel machine. If you use LAM and decompose your model in more domains than CPUs are available two or more domains will be updated on the same CPU (the program will not terminate and will produce the correct results). However, this is only efficient if more than one processor is available on each node. In order to reduce the amount of data that needs to be exchanged between PEs, you should decompose the model into more or less cubic sub grids. In our example, we use 2 PEs in each direction: $NPROCX=NPROCY=2$. The total number of PEs used by the program is $NPROC=NPROCX*NPROCY=4$.

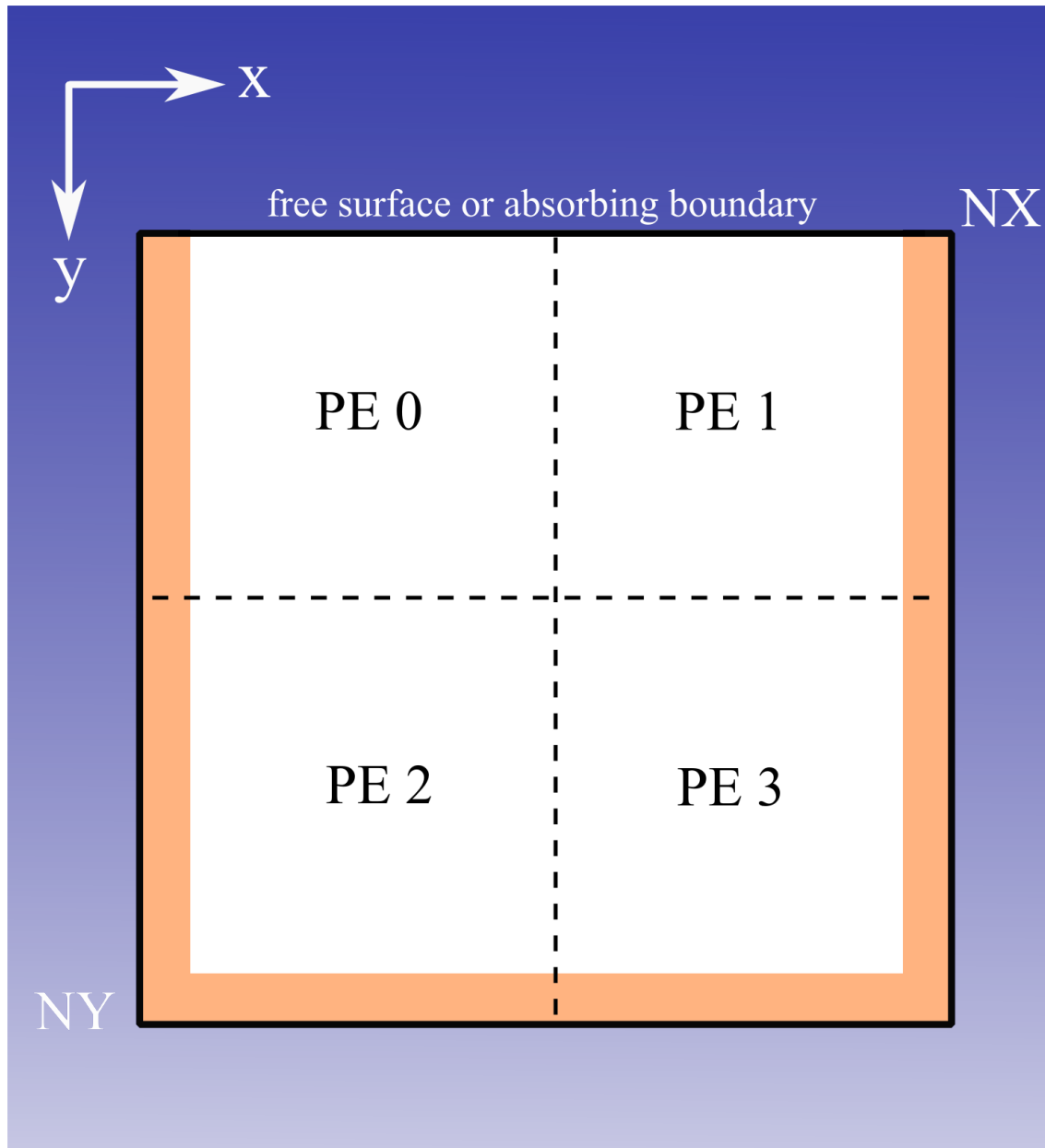


Figure 6.1: Geometry of the numerical FD grid using 2 processors in x -direction ($NPROCX=2$) and 2 processors in y -direction ($NPROCY=2$). Each processing element (PE) is updating the wavefield in its domain. At the top of the numerical mesh the PEs apply a free surface boundary condition if $FREE\_SURF=1$, otherwise an absorbing boundary condition (PML). The width of the absorbing frame is FW grid points. The size of the total grid is NX grid points in x -direction and NY gridpoints in y -direction. The size of each sub-grid thus is $NX/NPROCX \times NY/NPROCY$ gridpoints. The origin of the Cartesian coordinate system (x,y) is at the top left corner of the grid.

Order of the FD operator

```
#----- FD order -----
# Order of ssg FD coefficients (values: 2, 4, ..., 12)
FD_ORDER = 2
# Maximum relative group velocity error E
# (minimum number of grid points per shortest wavelength is defined by FD_ORDER)
# values: 0 = Taylor coefficients
#         1 = Holberg coeff.: E = 0.1 %
#         2 =                 E = 0.5 %
#         3 =                 E = 1.0 %
#         4 =                 E = 3.0 %
max_relative_error = 1
#
```

The order of the used FD operator is defined by the option FDORDER. With the option max_relative_error the user can switch between Taylor (max_relative_error=0) and Holberg (max_relative_error=1-4) FD coefficients of different accuracy. The chosen FD operator and FD coefficients have an influence on the numerical stability and grid dispersion (see chapter 2.3.1).

Discretization

```
#----- 2-D Grid -----
number_of_gridpoints_in_x-direction_(NX) = 800
number_of_gridpoints_in_y-direction_(NY) = 200
distance_between_gridpoints(in_m)_(DH) = 2.0e-4
#
```

These lines specify the size of the total numerical grid (Figure 6.1). NX and NY give the number of grid points in the x- and y-direction, respectively, and DH specify the grid spacing in x- and y-direction. The size of the total internal grid in meters in x-direction is NX*DH and in y-direction NY*DH. To allow for a consistent domain decomposition NX/NPROCX and NY/NPROCY must be integer values.

To avoid numerical dispersion the wavefield must be discretized with a certain number of gridpoints per wavelength. The number of gridpoints per wavelength required, depends on the order of the spatial FD operators used in the simulation (see section 2.3.1). In the current FD software, 2nd, 4th, 6th, 8th, 10th and 12th order operators are implemented. The criterion to avoid numerical dispersion is defined as:

$$DH \leq \frac{v_{s,min}}{2f_c n} \quad (6.1)$$

with $\frac{v_{s,min}}{2f_c}$ the smallest wavelength propagating through the model. $v_{s,min}$ denotes the minimum shear wave velocity in the model, and $f_c = 1/TS$ is the center frequency of the source wavelet. The program assumes that the maximum frequency of the source signal is approximately two times the center frequency. The center frequency is approximately one over the duration time TS. The value of n for different FD operators is tabulated in table 2.2. The criterion 6.1 is checked by the FD software. If the criterion is violated a warning message will be displayed in the DENISE output section “— CHECK FOR GRID DISPERSION —”. Please note, that the FD-code will NOT terminate due to grid dispersion, only a warning is given in the output file.

Time stepping

```
#-----Time Stepping -----
time_of_wave_propagation_(in_sec)_(TIME) = 1.8e-4
timestep_(in_seconds)_(DT) = 5.2e-8
#
```

The propagation time of seismic waves in the entire model is TIME. The time stepping interval (DT) has to fulfill the stability criterion (2.22) in section 2.3.2. The program checks these criteria for the entire model, outputs a warning message if these are violated, stops the program and will output the time step interval for a stable model run.

Sources

```
#-----Source-----
# Shape_of_source-signal:
(ricker=1;fumue=2;SOURCE_FILE=3;SIN*3=4;Gauss_deriv=5;Spike=6;Klauder=7)_(QUELLART) = 6
point_source_(explosive=1;force_in_x=2;force_in_y=3;rotated_force=4)_(QUELLTYP) = 3
angle_of_rotated_directed_source_relative_to_y_direction_(in_degree)_(ANGLE) = 0.0
depth_of_plane_wave_excitation_(no<=0)_(in_meter)_(PLANE_WAVE_DEPTH) = 0.0
dip_of_plane_wave_from_vertical_(in_degrees)_(PHI) = 0.0
SIGNAL_FILE = ./wavelet/wavelet_US_opt.dat
duration_of_Klauder_wavelet_(in_seconds)_(TS) = 8.0
read_source_positions_from_SOURCE_FILE_(yes=1)_(SRCREC) = 1
SOURCE_FILE = ./source/source.dat
run_multiple_shots_defined_in_SOURCE_FILE_(yes=1)_(RUN_MULTIPLE_SHOTS) = 1
corner_frequency_of_highpass_filtered_spike_(FC_SPIKE_1) = 5.0
corner_frequency_of_lowpass_filtered_spike_(FC_SPIKE_2) = 15.0
order_of_Butterworth_filter_(ORDER_SPIKE) = 5
#
```

6 built-in wavelets of the seismic source are available. The corresponding time functions are defined in src/wavelet.c. You may modify the time functions in this file and recompile to include your own analytical wavelet or to modify the shape of the built-in wavelets.

Ricker wavelet

$$r(\tau) = (1 - 2\tau^2) \exp(-\tau^2) \quad \text{with} \quad \tau = \frac{\pi(t - 1.5/f_c - t_d)}{1.0/f_c} \quad (6.2)$$

Fuchs-Müller wavelet

$$f_m(t) = \sin(2\pi(t - t_d)f_c) - 0.5 \sin(4\pi(t - t_d)f_c) \quad \text{if} \quad t \in [t_d, t_d + 1/f_c] \quad \text{else} \quad f_m(t) = 0 \quad (6.3)$$

sin³ wavelet

$$s_3(t) = 0.75\pi f_c \sin(\pi(t + t_d)f_c)^3 \quad \text{if} \quad t \in [t_d, t_d + 1/f_c] \quad \text{else} \quad s_3(t) = 0 \quad (6.4)$$

First Gaussian derivative wavelet

$$gd(t) = -2\pi^2 f_c^2 (t - t_d) \exp(-\pi^2 f_c^2 (t - t_d)^2) \quad (6.5)$$

Bandlimited spike wavelet

A spike, bandlimited by a low-pass or band-pass butterworth filter to avoid grid dispersion.

- If FC_SPIKE_1 <= 0.0 a low-pass filtered spike with upper corner frequency FC_SPIKE_2 and order ORDER_SPIKE is calculated
- If FC_SPIKE_1 > 0.0 a band-pass filtered spike with lower corner frequency FC_SPIKE_1 and upper corner frequency FC_SPIKE_2 with order ORDER_SPIKE is calculated

Klauder wavelet

A Klauder wavelet represents the autocorrelation of a linearly swept frequency-modulated sinusoidal signal used in Vibroseis

$$klau(t) = \text{real} \left\{ \sin \left(\frac{\pi k \tau (TS - \tau)}{\pi k \tau} \right) (\exp(2\pi i f_0 \tau)) \right\} \quad \text{with} \quad \tau = (t - 1.5/FC\_SPIKE\_1 - t_d) \quad (6.6)$$

with

$k = (FC\_SPIKE\_2 - FC\_SPIKE\_1)/TS$ (rate of change of frequency with time)

$f_0 = (FC\_SPIKE\_2 + FC\_SPIKE\_1)/2$ (midfrequency of bandwidth)

$i = \sqrt{-1}$

In these equations, t denotes time and f_c is the center frequency. t_d is a time delay which can be defined for each source position in `SOURCE_FILE`. Note that the symmetric (zero phase) Ricker signal is always delayed by $1.0/f_c$, which means that after one period the maximum amplitude is excited at the source location. These 5 source wavelets and the corresponding amplitude spectra for a center frequency of $f_c = 50$ Hz and a delay of $t_d = 0$ are plotted in Figure 6.2. Note the delay of the Ricker signal described above. The Fuchs-Müller wavelet has a slightly higher center frequency and covers a broader frequency range.

You may also use your own time function as the source wavelet (for instance the signal of the first arrival recorded by a geophone at near offsets). Specify `QUELLART=3` and save the samples of your source wavelet in ASCII-format in `SIGNAL_FILE`. `SIGNAL_FILE` should contain one sample per line. It should thus look like:

```
0.0
0.01
0.03
...
```

The time interval between the samples must equal the time step interval (DT) of the FD simulation (see above) ! Therefore it might be necessary to resample/interpolate a given source time function with a smaller sample rate. You may use the matlab script `mfiles/resamp.m` to resample your external source signal to the required sampling interval.

The following source types are available: explosive sources that excite compressional waves only (`QUELLTYP=1`), and point forces in the x- and y-direction (`QUELLTYP=2,3`). The force sources excite both P- and S-waves. The explosive source is located at the same position as the diagonal elements of the stress tensor, i.e. at (i,j) (Figure 2.1). The forces are located at the same position as the corresponding components of particle velocity (Figure 2.1). If (x,y) denotes the position at which the source location is defined in `source.dat`, then the actual force in x-direction is located at $(x+DX/2,y)$ and the actual force in y-direction is located at $(x,y+DY/2)$. With `QUELLTYP=4` a custom directive force can be defined by a force angle between y and x. The angle of the force must be specified in the `SOURCE_FILE` after `AMP`. This force is not aligned along the main directions. The parameter `ANGLE` is without any use.

The locations of multiple sources must be defined in an external ASCII file (`SOURCE_FILE`) that has the following format:

```
NSRC
%  XSRC ZSRC YSRC TD FC AMP  SOURCE_AZIMUTH SOURCE_TYPE (NSRC lines)
```

In the following lines, you can define certain parameters for each source point:

the first line must be the overall number of sources (NSRC). XSRC is the x-coordinate of a source point (in meter), YSRC is the y-coordinate of a source point (in meter). ZSRC is the z-coordinate should always be set to 0.0, because DENISE is a 2D code. TD is the excitation time (time-delay) for the source point [in seconds], FC is the center frequency of the source signal [in Hz], and AMP is the maximum amplitude of the source signal.

Optional parameter: The `SOURCE_AZIMUTH` if `SOURCE_TYPE` is 4. The `SOURCE_AZIMUTH` is the angle between the y- and x-direction and with `SOURCE_TYPE` if `SOURCE_TYPE` is set here, the value of `SOURCE_TYPE` in the input file is ignored.

The `SOURCE_FILE = ./sources/source.dat` that defines an explosive source at $x_s = 2592.0$ m and $y_s = 2106.0$ m with a center frequency of 5 Hz (no time delay) is

```
2592.0      0.0      2106.0      0.0      5.0      1.0      0.0
```

If the option `RUN_MULTIPLE_SHOTS=0` in the parameter file all shot points defined in the `SOURCE_FILE` are excited simultaneously in one simulation. Setting `RUN_MULTIPLE_SHOTS=1` will start individual model runs from $i=1$ to $i=NSRC$ with source locations and properties defined at line i of the `SOURCE_FILE`.

The options `PLANE_WAVE_DEPTH`, `TS` are obsolete and not supported in the current version of DENISE.

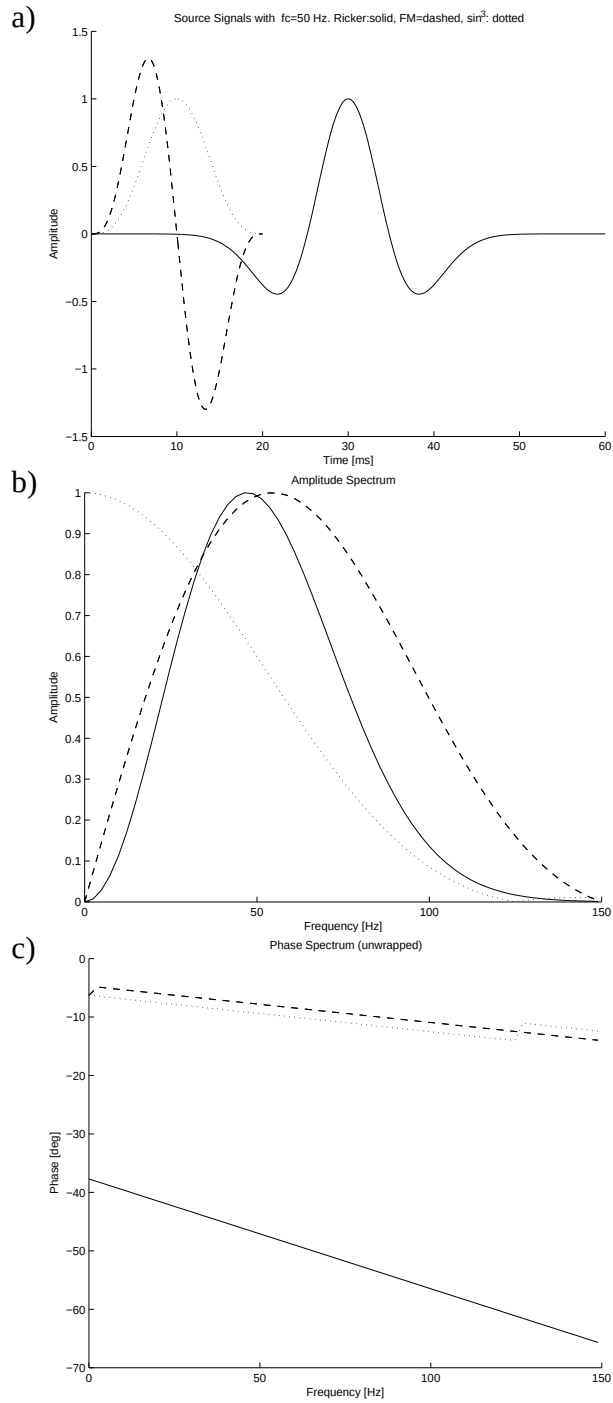


Figure 6.2: Plot of built-in source wavelets (equations 6.2, 6.3, 6.4) for a center frequency of $f_c = 50$ Hz ($TS = 1/f_c = 0.02$ s): Ricker signal (solid), Fuchs-Müller signal (dashed), $\sin^3$ -signal (dotted). a) Time function, b) amplitude spectrum, c) phase spectrum.

Model input

```
#----- Model -----
read_model_from_MFILE(yes=1) (READMOD) = 0
MFILE = model/test
#
```

If READMOD=1, the P-wave, S-wave, and density model grids are read from external binary files. MFILE defines the basic file name that is expanded by the following extensions: P-wave model: ".vp", S-wave model: ".vs", density model: ".rho". In the example above, the model files thus are: "model/test.vp" (P-wave velocity model), "model/test.vs" (S-wave velocity model), and "model/test.rho" (density model).

In these files, each material parameter value must be saved as 32 bit (4 byte) native float. Velocities must be in meter/second, density values in kg/m^3 . The fast dimension is the y direction. See src/readmod.c. The number of samples for the entire model in the x-direction is NX, the number of values in the y-direction is NY. The file size of each model file thus must be NX*NY*4 bytes. You may check the model structure using the SU command ximage: ximage n1=<NY> < model/test.vp .

It is also possible to read Qp, and Qs grid files to allow for spatial variable attenuation. For this you must uncomment a few lines in src/readmod.c and generate the corresponding binary files.

If READMOD=0 the model is generated "on the fly" by DENISE, i.e. it is generated internally before the time loop starts. If READMOD=0 this function is called in DENISE.c and therefore must be specified in src/Makefile (at the top of src/Makefile, see section 5.3). If you change this file, for example to change the model structure, you need to re-compile DENISE by changing to the src directory and "make denise".

Q-approximation

```
#-----Q-approximation-----
Number_of_relaxation_mechanisms_(L) = 0
L_Relaxation_frequencies_(FL) = 5.0
Tau_value_for_entire_model_(TAU0) = 0.00001
#
```

These lines may be used to define an overall level of intrinsic (viscoelastic) attenuation of seismic waves. In case of L=0, a purely elastic simulation is performed (no absorption). The frequency dependence of the (intrinsic) Quality factor $Q(\omega)$ is defined by the L relaxation frequencies ($FL=f_l = 2\pi/\tau_{\sigma l}$) and one value τ (see equation 5 in Bohlen [2002]). For a single relaxation mechanism (L=1) $Q \approx 2/\tau$ [Bohlen, 1998, Blanch et al., 1995, Bohlen, 2002]. If the model is generated "on the fly" the value of TAU can be assigned to all gridpoints for both P- and S-waves. Thus, intrinsic attenuation is homogeneous and equal for P- and S-waves ($Q_p(\omega) = Q_s(\omega)$). However, it is possible to simulate any spatial distribution of absorption by assigning the gridpoints with different Q-values by reading external grid files for Q_p (P-waves) and Q_s (S-waves) (see src/readmod.c) or by generating these files "on the fly" (see section ??).

Small Q values ($Q < 50$) may lead to significant amplitude decay and velocity dispersion. Please note, that due to dispersive media properties the viscoelastic velocity model is defined for the reference frequency only. In denise, this reference frequency is specified as the center source frequency. At the exact reference frequency, elastic and viscoelastic models are equivalent. As a consequence, slightly smaller and larger minimum and maximum velocity values occur in the viscoelastic model.

The frequency dependence of attenuation, i.e. Q and phase velocity as a function of frequency, may be calculated using the Matlab functions in the directory mfiles.

Free surface

```
#-----Free Surface-----
free_surface_(yes=1) (FREE_SURF) = 1
```

A plane stress free surface is applied at the top of the global grid if `FREE_SURF!=0` using the imaging method proposed by Levander [1988]. Note that **the free surface is always located at $y=0$** .

Boundary conditions

```
#-----PML Boundary-----
# quadratic damping applied
width_of_absorbing_frame_(in_gridpoints)_(No<=0)_(FW) = 10
Damping_velocity_in_CPML_(in_m/s)_(DAMPING) = 1277.0
Frequency_within_the_PML_(Hz)_(FPML) = 100.0
npower = 4.0
k_max_PML = 1.0
# apply periodic boundary condition at edges_(BOUNDARY):
(no=0)_(left_and_right=1) = 0
#
```

The boundary conditions are applied on each side face and the bottom face of the model grid. If `FREE_SURF = 0` the boundary conditions are also applied at the top face of the model grid. Note that the absorbing frames are always located **INSIDE** the model space, i.e. parts of the model structure are covered by the absorbing frame, in which no physically meaningful wavefield propagates. You should therefore consider the frame width when you design the model structure and the acquisition geometry (shot and receivers should certainly be placed outside).

A convolutional perfectly matched layer (CPML) boundary condition is used. The PML implementation is based on the following papers Komatitsch and Martin [2007]. A width of the absorbing frame of `FW=10-20` grid points should be sufficient. For the optimal realization of the PML boundary condition you have to specify the dominant signal frequency `FPML` occurring during the wave simulation. This is usually the center source frequency `FC` specified in the source file. `DAMPING` specifies the attenuation velocity in m/s within the PML. `DAMPING` should be approximately the propagation velocity of the dominant wave near the model boundaries.

In some cases, it is useful to apply periodic boundary conditions (see section 2.2.3). If `BOUNDARY=1` no absorbing boundaries are installed at the left/right sides of the grid. Instead, wavefield information is copied from left to right and vice versa. Therefore a wave which leaves the model at the left side enters the model again at the right side and vice versa.

Wavefield snapshots

```
#-----Snapshots-----
output_of_snapshots_(SNAP) (yes>0) = 0
# output of particle velocities: SNAP=1
# output of pressure field: SNAP=2
# output of curl and divergence energy: SNAP=3
# output of both particle velocities and energy : SNAP=4
first_snapshot_(in_sec)_(TSNAP1) = 3e-6
last_snapshot_(in_sec)_(TSNAP2) = 1.5e-4
increment_(in_sec)_(TSNAPINC) = 3e-6
increment_x-direction_(IDX) = 1
increment_y-direction_(IDY) = 1
data-format_(SNAP_FORMAT) (ASCII(2);BINARY(3)) = 3
basic_filename_(SNAP_FILE) = ./snap/waveform_forward
#
```

If `SNAP> 0`, wavefield information (particle velocities, pressure, or curl and divergence of particle velocities) for the entire model is saved on the hard disk (assure that enough free space is on disk!). Each PE is writing his sub-volume to disk. The filenames have the basic filename `SNAP_FILE` plus an extension that indicates the PE number in the logical processor array (see Figure 6.1), i.e. the PE with number `PEno` writes his wavefield to `SNAP-FILE.PEno`. The first snapshot is written at `TSNAP1` seconds of seismic wave travelttime to the output files, the second

at TSNAP1+TSNAPINC seconds etc. The last snapshots contains wavefield at TSNAP2 seconds. Note that the file sizes increase during the simulation. The snapshot files might become quite LARGE. It may therefore be necessary to reduce the amount of snapshot data by increasing IDX, IDY and/or TSNAPINC. In order to merge the separate snapshot of each PE after the completion of the wave modeling, you can use the program snapmerge (see Chapter 5.2, section **src**). The bash command line to merge the snapshot files can look like this: `../bin/snapmerge DENISE.inp`.

Receivers

```
#-----Receiver-----
output_of_seismograms_(SEISMO) = 1
# SEISMO=0: no seismograms
# SEISMO=1: particle-velocities
# SEISMO=2: pressure (hydrophones)
# SEISMO=3: curl and div
# SEISMO=4: everything
read_receiver_positions_from_file_(yes=1)_(READREC) = 1
REC_FILE = ./receiver/receiver_us.dat
reference_point_for_receiver_coordinate_system_(REFREC) = 0.0 , 0.0
# if READREC=1 the following three lines are ignored
position_of_first_receiver_(in_m)_(XREC1,YREC1) = 690.0 , 2300.0
position_of_last_receiver_(in_m)_(XREC2,YREC2) = 790.0 , 300.0
distance_between_two_adjacent_receivers_(in_gridpoints)_(NGEOPH) = 6
#
```

If SEISMO>0, seismograms are saved on hard disk. If SEISMO equals 1 x- and y-component of particle velocity will be written according to parameters specified in Chapter 6.1. If SEISMO==2 pressure (sum of the diagonal components of the stress tensor) recorded at the receiver locations (receivers are hydrophones!) is written. if SEISMO=3 the curl and divergence are saved.

The curl and divergence of the particle velocities are useful to separate between P- and S-waves in the snapshots of the wavefield. DENISE calculates the divergence and the magnitude of the curl of the particle velocity field according to Dougherty and Stephen [1988]. The motivation for this is as follows. According to Morse and Feshbach Morse and Feshbach [1953] the energy of P- and S-wave particle velocities is, respectively,

$$E_p = (\lambda + 2\mu) (div(\vec{v}))^2 \quad \text{and} \quad E_s = \mu |rot(\vec{v})|^2 . \quad (6.7)$$

λ and μ are the Lamè parameters, and $\vec{v}$ is the particle velocity vector.

The locations of the receivers may either be specified in a separate file REC_FILE or in this parameter file. If READREC=1 receiver locations are read from the ASCII-file REC_FILE. Each line contains the coordinates of one receiver, the first two number in each line indicate the horizontal x- and the vertical y-coordinate of each receiver position. To give an example of a receiver file, the following 3 lines specify 3 receivers located at constant depth (2106.0 m). However, the receiver coordinates change in x-direction (starting at 540 m) and therefore lining up along the x-axis.

```
540.0   2106.0
1080.0  2106.0
1620.0  2106.0
```

These receiver coordinates in REC_FILE are shifted by REFREC[1], REFREC[2] into the x- and y-direction, respectively. This allows for completely moving the receiver spread without modifying REC_FILE. This may be useful for the simulation of moving profiles in reflection seismics.

If READREC=0 the receiver locations must be specified in the parameter file. In this case, it is assumed that the receivers are located along a straight line. The first receiver position is defined by (XREC1, YREC1), and the last receiver position by (XREC2, YREC2) (see Figure 6.1). The spacing between receivers is NGEOPH grid points.

Receivers are always located on full grid indices, i.e. a receiver that is located between two grid points will be shifted by the FD program to the closest next grid point. It is not yet possible to output seismograms for arbitrary receiver locations since this would require a certain wavefield interpolation.

It is important to note that the actual receiver positions defined in REC_FILE or in DENISE.inp may vary by $DX/2$ and/or $DY/2$ and/or $DZ/2$ due to the staggered positions of the particle velocities and stress tensor components.

In our example, we specify 100 receiver location. Due to the small size of the model, most of the specified receiver positions will be located inside this absorbing boundary (if $ABS=2$, see Chapter 6.1). A corresponding warning message will appear. If you choose to read the receiver location from REC_FILE ./receiver/receiver.dat (READREC=1), only 10 receivers locations are considered. The list of receivers specified in file ./receiver/receiver.dat is equivalent to the parameters in the input file with only a larger distance between adjacent receivers (NGEOPH = 10.)

Towed streamer

```
#----- Towed streamer -----
# parameters for towed streamer acquisition
The_first_(N_STREAMER)_receivers_in_REC_FILE_belong_to_streamer = 31
Cable_increment_per_shot_(REC_INCR_X) = 80.0
Cable_increment_per_shot_(REC_INCR_Y) = 0
#
```

A streamer geometry can be defined by moving the receiver positions, defined in the receiver file REC_FILE, in accordance with the sources in x-direction by REC_INCR_X or y-direction by REC_INCR_Y. In this example the receivers are moved in x-direction by 80 m for each shot increment. More complex acquisition geometries, e.g. streamer and OBS, can be implemented by assuming, that only the first N_STREAMER receivers contribute to the streamer and are subsequently moved. All other receiver positions in the receiver file REC_FILE are fixed.

Seismograms

```
#----- Seismograms -----
samplingrate_(in_timesteps!)(NDT) = 1
data-format_(SU(1);ASCII(2);BINARY(3)) = 1
# output files for seismograms
# particle velocities (if SEISMO=1 or SEISMO=4)
filename_for_Vx_(SEIS_FILE_VX) = su/DENISE_US_x.su
filename_for_Vy_(SEIS_FILE_VY) = su/DENISE_US_y.su
# curl and div of wavefield (if SEISMO=3 or SEISMO=4)
filename_for_curl_(SEIS_FILE_CURL) = su/2layer_rot.su
filename_for_div_(SEIS_FILE_DIV) = su/2layer_div.su
# pressure field (hydrophones) (if SEISMO=2 or SEISMO=4)
filename_for_pressure_(SEIS_FILE_P) = su/full_wave_forward_p.su
#
```

If $SEISMO > 0$ seismograms recorded at the receiver positions are written to the corresponding output files. The sampling rate of the seismograms is $NDT \cdot DT$ seconds. In case of a small time step interval and a high number of time steps, it might be useful to choose a high NDT in order to avoid a unnecessary detailed sampling of the seismograms and consequently large files of seismogram data. Keep in mind that the application of FWT requires $NDT=1$. Possible output formats of the seismograms are SU, ASCII and BINARY. It is recommended to use SU format for saving the seismograms. The main advantage of this format is that the time step interval ($NDT \cdot DT$) and the acquisition geometry (shot and receiver locations) are stored in the corresponding SU header words. Also additional header words like offset are set by DENISE. This format thus facilitates a further visualization and processing of the synthetic seismograms. Note, however, that SU cannot handle sampling rates smaller than $1.0e-6$ seconds and the number of samples is limited to about 32.000. In such cases, you should increase the sampling rate by increasing NDT. If this is impossible (for example because the Nyquist criterion is violated) you must choose a different output format (ASCII or binary).

Monitoring the simulation

```
# each PE is printing log-information to LOG_FILE.MYID
log-file_for_information_about_progress_of_program_(LOG_FILE) = log/2layer.log
info_of_processing_element_zero_to_stdout_(yes=1/no=0)_(LOG) = 1
```

DENISE can output a lot of useful information about the modeling parameters and the status of the modeling process etc. The major part of this information is output by PE 0. If LOG=1, PE 0 writes this info to stdout, i.e. on the screen of your shell. This is generally recommended to monitor the modeling process. You may want to save this screen info to an output file by adding "> DENISE.out" or "tee DENISE.out". to your starting command. If LOG=1 all other processes with PE number (PEno) greater than zero will write their information to LOG_FILE.PEno. If you specify LOG=2 PE 0 will also output information to LOG_FILE.0. As a consequence only little information is written directly to the screen of your shell. On supercomputers where you submit modeling jobs to a queuing system as batch jobs LOG=2 may be advantageous. In case of LOG=2, you may still watch the simulation by checking the content of LOG_FILE.0 for example by using the Unix commands more or tail. After finishing the program the timing information is written to the ASCII file log/test.log.0.timings. This feature is useful to benchmark your local PC cluster or supercomputer. If LOG=0 no output from node 0 will be written, neither to stdout nor to an LOG file. There will be also no output of timing information to the ASCII file log/test.log.0.timings.

Checkpointing

```
#-----Checkpoints -----
read_wavefield_from_checkpoint_file_(yes=1/no=0)_(CHECKPTREAD) = 0
save_wavefield_to_checkpoint_file_(yes=1/no=0)_(CHECKPTWRITE) = 0
checkpoint_file_(CHECKPTFILE) = tmp/checkpoint_DENISE
```

These options are obsolete and are not supported in the current version of DENISE.

General inversion parameters

```
#----- General DENISE inversion parameters -----
number_of_TDFWI_iterations_(ITERMAX) = 100
output_of_jacobian_(JACOBIAN) = jacobian/jacobian_Test
seismograms_of_measured_data_(DATA_DIR) = su/plexiglas/DENISE_plexiglas
cosine_taper_(yes=1/no=0)_(TAPER) = 0
taper_length_(in_rec_numbers)_(TAPERLENGTH) = 5
Inverse_Type_(gradient=1/complete=2)_(INVTYPE) = 2
gradient_taper_geometry_(GRADT1,GRADT2,GRADT3,GRADT4) = 5, 10, 510, 520
type_of_material_parameters_(Vp,Vs,rho=1/Zp,Zs,rho=2/lam,mu,rho=3)_(INVMAT1) = 1
forward_modelling_only_(yes=10)_FWT_(yes=0)_(INVMAT) = 0
point_source_backpropagation_(x_and_y_comp=1/y_comp=2/x_comp=3/p_comp=4)_(QUELLTYPB) = 2
testshots_for_step_length_est_(TESTSHOT_START,TESTSHOT_END,TESTSHOT_INCR) = 1,21,10
#
```

This section covers some general inversion parameters. The maximum number of iterations are defined by ITERMAX. The switch INVMAT controls if only the forward modeling code should be used (INVMAT=10), e.g. to calculate synthetic seismograms or a complete FWT run (INVMAT=0). In case of INVMAT=10 the parameters in the workflow file (section 6.2) are ignored, but a workflow file still has to be defined. The seismic sections of the real field data are located in the DATA_DIR. As noted in section 3.4 the gradients can be expressed for different model parameterizations. The switch INVMAT1 defines which parameterization should be used, seismic velocities and density (Vp,Vs,rho, INVMAT1=1), seismic impedances (Zp,Zs,rho, INVMAT1=2) or Lamé parameters (λ, μ, ρ , INVMAT1=3). If models are read from binary files appropriate file extensions are required for the different models (see section 6.1). Depending on the data different components of the seismic sections can be backpropagated. For two

component data (x- and y-component) set QUELLTYPB=1, y-component only (QUELLTYPB=2), x-component only (QUELLTYPB=3) and pressure component (QUELLTYPB=4). To estimate the step length μ for the model update requires the calculation of the misfit function and therefore the solution of the forward problem for all shots involved. However, to save computation time the calculation of the misfit function can be restricted to certain shots, using only shots between TESTSHOT_START and TESTSHOT_END with a shot increment TESTSHOT_INCR.

The parameters TAPER, TAPERLENGTH and INVTYPE are debug parameters and should not be changed. The parameters JACOBIAN and GRADT1 to GRADT4 are explained in section 6.1 and section 6.1 respectively.

Definition of the gradient taper geometry

```
# ----- Definition of gradient taper geometry ----- #
# Vertical taper
apply_vertical_taper_(yes=1)_(SWS_TAPER_GRAD_VERT) = 0
# Horizontal taper
apply_horizontal_taper_(yes=1)_(SWS_TAPER_GRAD_HOR) = 0
exponent_of_depth_scaling_for_preconditioning_(EXP_TAPER_GRAD_HOR) = 2.0
# Circular taper around all sources (not at receiver positions)
apply_cylindrical_taper_(yes=1)_(SWS_TAPER_GRAD_SOURCES) = 0
apply_cylindrical_taper_per_shot_(yes=1)_(SWS_TAPER_CIRCULAR_PER_SHOT) = 1
(1=error_function,2=log_function)_(SRTSHAPE) = 1
radius_in_m_(SRTRADIUS) = 5e-3
# --> minimum for SRTRADIUS is 5x5 gridpoints
filtsize_in_gridpoints_(FILTSIZE) = 25
read_taper_from_file_(yes=1)_(SWS_TAPER_FILE) = 0
#
```

Different preconditioning operators can be created and applied to the gradients (using the function 'taper_grad.c') to improve convergence speed, depth resolution and define which parts of the model should be updated. To apply a vertical or a horizontal taper one has to set the switches SWS_TAPER_GRAD_VERT and SWS_TAPER_GRAD_HOR to 1, respectively. The parameters for the vertical and the horizontal window are defined by the input file parameters GRADT1, GRADT2, GRADT3 and GRADT4. Please have a look at the function taper_grad.c directly to obtain more information about the actual definition of the tapers. In case of SWS_TAPER_GRAD_HOR=1 the gradient can also be scaled with the following depth dependent preconditioning operator:

$$P = \text{depth}^{\text{EXP\_TAPER\_GRAD\_HOR}} \quad (6.8)$$

It is also possible to apply cylindrical tapers around the source positions. This can be done by setting the switch SWS_TAPER_GRAD_SOURCES or SWS_TAPER_CIRCULAR_PER_SHOT to 1. If one uses SWS_TAPER_GRAD_SOURCES=1 only the final gradients (that means the gradients obtained by the summation of the gradients of each shots) are multiplied with a taper that decreases the gradients at all shot positions. Therefore, one loses the update information at the source positions. To avoid this one can use SWS_TAPER_CIRCULAR_PER_SHOT=1. In this case the gradients of the single shots are preconditioned with a window that only damps the gradient at the current shot position. This is done before the summation of all gradients to keep model update information at the shot positions. The actual tapers are generated by the function 'taper_grad.c' and 'taper_grad_shot.c', respectively. The circular taper around the source positions decrease from a value of one at the edge of the taper to a value of zero at the source position. The damping shape can be defined by an error function (SRTSHAPE=1) or a log-function (SRTSHAPE=2). The radius of the taper is defined in meter by SRTRADIUS. Note, that this radius must be at least 6 gridpoints. With the parameter FILTSIZE one can extend the region where the taper is zero around the source. The taper is set to zero in a square region of (2*FILTSIZE+1 times 2*FILTSIZE+1) gridpoints. All preconditioning matrices applied to the gradients are saved in the par directory with the file names taper_coeff_vert.bin, taper_coeff_horz.bin and taper_coeff_sources.bin. To apply an externally defined taper on the gradients in DENISE, the parameter SWS_TAPER_FILE has to be set to 1. Each model parameter requires a taper file which should be located in the /par directory and named as taper.bin for the Vp-model, taper_u.bin for the Vs-model and taper_rho.bin for the density model.

Output of inversion results

```
# ----- Output of inverted models ----- #
output_of_models_(INV_MODELFILE) = model/model_Test
first_model_to_be_saved_(nfstart) = 2000
increment_between_saved_models_(nf) = 2000
#
#
# ----- Output of gradients ----- #
first_gradient_to_be_saved(nfstart_jac) = 2000
increment_between_saved_gradients_(nf_jac) = 1
#
```

Options can produce conflicts in the current version of DENISE. It is highly recommended to use the default values.

Limits for the model parameters

```
# ----- Upper and lower limits for model parameters ----- #
upper_limit_for_vp/lambda_(VPUPPERLIM) = 2600
lower_limit_for_vp/lambda_(VPLOWERLIM) = 500
upper_limit_for_vs/mu_(VSUPPERLIM) = 1600
lower_limit_for_vs/mu_(VSLOWERLIM) = 1000
upper_limit_for_rho_(RHOUPPERLIM) = 5000
lower_limit_for_rho_(RHOLOWERLIM) = 0
#
```

The six limits for the model parameters specify the minimum and maximum values which may be achieved by the elastic inversion. Here, known a priori information can be used. Depending on the choice of the parameter INVMAT1, either vp and vs or lambda and mu is limited.

Optimization method

```
# ----- Optimization-Method ----- #
gradient_method_(PCG=1/LBFGS=2/GRADIENT=3)_(GRAD_METHOD) = 2
save_(NLBFGS)_updates_during_LBFGS_optimization = 20
#
```

During FWI the misfit function can be minimized by different optimization methods. Currently a preconditioned conjugate gradient (PCG), the quasi-Newton method limited memory Broyden-Fletcher-Goldfarb-Shanno l-BFGS method (see e.g. Nocedal and Wright [2006]) and a simple gradient method can be applied. When using the l-BFGS optimization the last NLBFGS updates are stored.

Reduce inversion grid

```
#----- Reduce inversion grid -----#
use_only_every_DTINV_time_sample_for_gradient_calculation_(DTINV) = 3
#
```

To reduce the memory requirements during an inversion one can define that only every DTINV time sample is used for the calculation of the gradients. To set this parameter appropriately one has to keep in mind the Nyquist criterion to avoid aliasing effects.

Step length estimation

```
#----- Step length estimation -----#
maximum_model_change_of_maximum_model_value_(EPS_SCALE) = 0.01
maximum_number_of_attempts_to_find_a_step_length_(STEPMAX) = 4
SCALEFAC = 4.0
testshots_for_step_length_est_(TESTSHOT_START,TESTSHOT_END,TESTSHOT_INCR) = 1, 1, 1
#
```

For the step length estimation a parabolic line search method proposed by Sourbier et al. [2009a,b], Brossier [2009] and Nocedal and Wright [2006] is implemented. For this step length estimation only two further test forward modelings are needed. The vector `L2t` contains the misfit values and the vector `epst` contains the corresponding step length. During the forward modeling of an iteration step the misfit norm of the data residuals is calculated for the shots defined by `TESTSHOT_START`, `TESTSHOT_END` and `TESTSHOT_INC`. The value `L2t[1]` then contains the misfit from the forward modeling and the corresponding `epst[1]` value is 0.0.

The step lengths for the different parameters are defined as:
 $\text{EPSILON} = \text{EPS\_SCALE} * \text{m\_max/grad\_max}$ $\text{EPSILON} = \text{epst}[i] * \text{m\_max/grad\_max}$
 where `m_max` is the maximum value of the corresponding model parameter in the whole model and `grad_max` is the maximum absolute value of the gradient.

For a better definition of the parabola the improved line search is now trying to estimate a steplength `epst[2]` with $\text{L2t}[2] < \text{L2t}[1]$. If the code is not able to find an appropriate steplength using the user-defined value `EPS_SCALE` (f.e. $\text{EPS\_SCALE} = 0.01 = 1\%$ change in terms of `m_max/grad_max`), the code divides this steplength by the variable `SCALEFAC` and calculates the misfit norm again. If this search fails after `STEPMAX` attempts `DENISE` exits with an error message. If the algorithm has found an appropriate value for `epst[2]`, it is trying to estimate a steplength `epst[3]` with $\text{L2t}[3] > \text{L2t}[2]$, by increasing the steplength

$\text{EPS\_SCALE} += \text{EPS\_SCALE}/\text{SCALEFAC}.$

If a corresponding value `epst[3]` can be found after `STEPMAX` forward modellings, `DENISE` can fit a parabola through the 3 points (`L2t[i]`, `epst[i]`) and estimates an optimum step length at the minimum of the parabola. If the `L2t`-value `L2t[3]` after `STEPMAX` forward models is still smaller than `L2t[2]` the optimum steplength estimated by parabolic fitting will be not larger than `epst[3]`.

Trace killing

```
#----- Trace killing -----#
apply_trace_killing_(yes=1)_(TRKILL) = 0
TRKILL_FILE = ./trace_kill/trace_kill.dat
```

To mute noisy or unwanted traces during FWI, the parameter `TRKILL` is introduced. If `TRKILL` is set to 1, all traces defined in the parameter file `TRKILL_FILE` are muted. The file should include a mute table, where the rows have the same lengths as the number of receivers and the columns reflects the number of sources. A 1 (ONE) indicates a mute of the trace, while a 0 (ZERO) means that this trace should NOT be muted.

Time damping

```
#----- Time windowing and damping -----#
files_with_picked_times_(PICKS_FILE) = ./picked_times/picks_
```

If time damping of the seismograms is activated in the workflow file by setting `TIMWIN!=0` picked times of specific seismic phases, like first arrivals, for each source and receiver must be specified in a separate file. The folder and

file name can be set with the parameter PICKS_FILE. The files must be named like this [PICKS_FILE]_sourcenum.dat. So the number of sources in (SRCREC) must be equal to the number of files. Each file must contain the picked times for every receiver. Other important parameters are set in the workflow file (see section 6.2).

Name of the misfit log file

```
#----- MISFIT LOG FILE -----#
log_file_for_misfit_evolution_(MISFIT_LOG_FILE) = LOG_TEST.dat
#
```

The name of the misfit log file can be changed with the parameter MISFIT_LOG_FILE. The columns of the misfit log file contain information about the step length and misfit function values acquired during the step length estimation and the stage number nstage:

```
opteps epst[1] epst[2] epst[3] L2t[1] L2t[2] L2t[3] L2t[1] nstage
```

When a frequency filter is applied information about the corner frequencies are also written to the misfit log file.

```
opteps epst[1] epst[2] epst[3] L2t[1] L2t[2] L2t[3] L2t[1] FC_low FC_high nstage
```

2D Gaussian gradient smoothing filter

```
# ----- Definition of smoothing the Jacobians with 2D-Gaussian ----- #
apply_spatial_filtering_(yes=1)_(GRAD_FILTER) = 0
filter_length_in_gridpoints_(FILT_SIZE_GRAD) = 10
#
```

If GRAD_FILTER=1 smooth the gradients for the different material parameters using a 2D-Gaussian with a filter-length defined by FILT_SIZE_GRAD in gridpoints.

Time-lapse FWI mode

```
# ----- FWT double-difference time-lapse mode ----- #
activate_time_lapse_mode_(yes=1)_(TIMELAPSE) = 0
# if TIMELAPSE == 1, DATA_DIR should be the directory containing the data differences
# between time t0 and t1
seismograms_of_synthetic_data_at_t0_(DATA_DIR_T0) = su/CAES_spike_time_0/DENISE_CAES
#
```

If TIMELAPSE=1 the spatial FWI is replaced by a double-difference time-lapse FWI Denli and Huang [2009], al Hagrey et al. [2014]. In this case DATA_DIR defines the data differences between the baseline data at time t0 and the time-lapse data at t1. For existing SU-files the data differences can be calculated with the shell script *time_lapse_data_diff.sh* in the par-directory. The location of the baseline data can be defined with DATA_DIR_T0.

Elastic Reverse Time Migration

```
# ----- Elastic Reverse Time Migration ----- #
apply_reverse_time_migration_(yes=1)_(RTM) = 0
#
```

If RTM=1 and INVMAT=0 an elastic Reverse Time Migration (RTM) is applied for the field data defined in the directory DATA_DIR. If time-lapse mode is activated (TIMELAPSE=1) the time-lapse data will be migrated. The workflow file (section 6.2) should only contain one stage. The resulting migrated seismic sections are written to the directory JACOBIAN. Currently Reverse Time Migration is only defined for the L2-Norm (L2NORM=2).

6.2 Workflow file with variable inversion parameters FWI_workflow.inp

Complex FWI workflows can be designed with the input file shown in table 7.2. Each line represents a FWI stage with a specific combination of different inversion parameters, defined in the columns.

Abort criterion

Beside the parameter ITERMAX a second abort criterion is implemented in DENISE which is using the relative misfit change within the last two iterations. The relative misfit of the current iteration step and the misfit of the second to last iteration step is calculated with regard to the misfit of the second to last iteration step. If this relative change is smaller than PRO the inversion aborts or proceeds to the next inversion stage.

Frequency filtering

To tame the nonlinearity of the inversion problem Butterworth frequency filters can be applied to the source wavelet and field data.

- TIME_FILT=1 applies a lowpass frequency filter with an upper corner frequency FC_high.
- TIME_FILT=2 applies a bandpass frequency filter with a lower corner frequency FC_low and upper corner frequency FC_high.

The order of the Butterworth filter is defined by the parameter ORDER.

Time damping

Multiple or complex reflections can significantly increase the nonlinearity of the inverse problem. Different time-damping strategies are implemented in DENISE to

- TIMEWIN=1 reads traveltimes picks of the first arrival from the PICKS_FILES defined in the parameter file (section 6). A constant time-delay TWIN+ can be applied to each pick.
- TIMEWIN=2 applies a time-damping from a constant time TWIN+ for all receivers and shots.

The amount of damping can be defined by the parameter GAMMA.

Definition of multiparameter inversion

The parameters INV_VP_ITER, INV_VS_ITER, INV_RHO_ITER define from which iteration step the corresponding parameters V_p , V_s and ρ are updated. Setting a parameter to 0 updates the model during all iterations, while setting a parameter larger than ITERMAX no model update is applied. Combinations of these parameters allows the implementation of a simultaneous or hierarchical inversion workflow.

Spatial filtering of gradients

To suppress short wavelength artefacts below the source and receiver positions the gradients can be smoothed.

- SPATFILTER=1 applies a wavenumber domain damping with a Gaussian function

$$\hat{g}(k_x, k_y) = g(k_x, k_y) \exp(-WD\_DAMP(k_x^2 + k_y^2))$$

to the gradients $g(k_x, k_y)$. The amount of damping can be controlled by the parameter WD_DAMP.

- SPATFILTER=2 applies a damped least squares technique to the gradients. The size of the filter is defined by WD_DAMP in y-direction and WD_DAMP1 in x-direction.

Preconditioning

To accelerate the convergence speed of the optimization method and avoid the convergence in a local minimum, amplitude loss with depth due to geometrical spreading and reflections in the upper model parts have to be compensated. In case of Quasi-Newton or Full-Newton methods these effects are corrected by the inverse Hessian. For (conjugate) gradient methods different approximations of the inverse Hessian can be used as preconditioning operator.

- EPRECOND=1 approximates the inverse of the Hessian by the absolute value of the forward wavefield [Shin et al., 2001]:

$$H_a^{-1} = \left\{ \int dt |\mathbf{u}(\mathbf{x}_s, \mathbf{x}, t)|^2 \right\}^{-1}.$$

- EPRECOND=3 approximates the inverse of the Hessian by a zero-lag correlation of the absolute value of the forward wavefield with an approximation of the receiver Greens function contribution [Plessix and Mulder, 2004]:

$$H_a^{-1} = \left\{ \int dt |\mathbf{u}(\mathbf{x}_s, \mathbf{x}, t)|^2 \left[\operatorname{asinh}\left(\frac{x_r^{\max} - x}{z}\right) - \operatorname{asinh}\left(\frac{x_r^{\min} - x}{z}\right) \right] \right\}^{-1},$$

where $x_r^{\min}$, $x_r^{\max}$, $\mathbf{x}_s$ denote the minimum, maximum receiver and source positions.

Misfit definition

Different objective functions can have a significant impact on the nonlinearity of the inverse problem. Changing the misfit function between the modelled data $\mathbf{u}$ and field data $\mathbf{d}$ does only change the backpropagated residuals in the FWI algorithm.

- LNORM=2 sets the misfit function to the "classical" L2 norm of the data residuals Eq. (3.3)

$$E_{L2} = \frac{1}{2} \sum_i^{ns} \sum_j^{nr} (\mathbf{u}_{ij} - \mathbf{d}_{ij})^2.$$

In this case the misfit is scaled with the energy of the measured seismograms.

- LNORM=5 sets the misfit function to the global correlation norm [Choi and Alkhalifah, 2012]

$$E_{GC} = - \sum_i^{ns} \sum_j^{nr} \left[\frac{\mathbf{u}_{ij}}{\|\mathbf{u}_{ij}\|} \cdot \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|} \right]$$

Source wavelet inversion

As discussed in chapter 4 the estimation of the source wavelet is vital for a successful FWI. With STF_INV=1 a source wavelet inversion by a stabilized Wiener deconvolution is activated. This requires one additional forward model run per shot to estimate the Greens function solution for the actual model. The parameter NTR_STF defines how many traces in the vicinity of the shot point are used. In case of dispersive wavefields it is recommended to limit the source wavelet inversion only to the near-offset traces to avoid the interpretation of model parameter changes as source wavelet. Traces with maximum offsets OFFSETC_STF are used for the wavelet inversion. EPS_STF denotes the regularization parameter defined in Eq. (4.6). So far the source wavelet will be only estimated from the vertical component data. In case of STF_INV=2 the first arrival of the synthetic data will be automatically picked by an STA/LTA picker and the amplitudes of the field data before the first arrival and after the first arrival plus a time window defined by TW_STF are exponentially damped. The amount of damping is controlled by GAMMA_STF. TW_STF and GAMMA_STF currently have to be set in time_window_stf.c.

Normalization

Normalize seismic data and source wavelet during source wavelet inversion. This can be required if the amplitudes of source wavelets for each shot show strong variations. The normalization leads to an equalization of the shot contributions to the gradient.

- NORMALIZE=0 No normalization of seismic data and source wavelet during source wavelet inversion.
- NORMALIZE=1 Trace normalization of synthetic and field data to the maximum amplitude of each trace. Normalize the estimated source wavelet to its maximum amplitude.
- NORMALIZE=2 Normalize for each trace the maximum of the field data relative to the maximum of the synthetic data. Normalize the estimated source wavelet to its maximum amplitude.

Offset-Windowing

In some cases the application of an offset-window can be useful to achieve a "layer-stripping" update of the model parameters from top to bottom.

- OFFSET_MUTE=1 mutes all traces with an offset larger than OFFSETC (far-offset mute).
- OFFSET_MUTE=2 mutes all traces with an offset smaller than OFFSETC (near-offset mute).

Density model update restriction

Because changes of the density model are in most cases smaller than velocity changes the step length for the density update can be systematically reduced by a factor SCALERHO (see Eq. (3.49)).

PRO	TIME_FILT	FC_low	FC_high	ORDER	TIME_WIN	GAMMA	TWIN-	TWIN+	INV_VP_ITER	INV_VS_ITER	INV_RHO_ITER	SPATFILTER	WD_DAMP	WD_DAMP1	EPRECOND	LNORM	STF_INV	OFFSETC_STF	EPS_STF	NORMALIZE	OFFSET_MUTE	OFFSETC	SCALERHO
0.01	1	0.0	1.7	6	2	1e2	0.0	1.75	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	2.9	6	2	1e2	0.0	1.75	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	1.75	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	1.7	6	2	1e2	0.0	2.55	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	2.9	6	2	1e2	0.0	2.55	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	2.55	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	3.55	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	5.15	0	600	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	1.7	6	2	1e2	0.0	1.75	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	2.9	6	2	1e2	0.0	1.75	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	1.75	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	1.7	6	2	1e2	0.0	2.55	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	2.9	6	2	1e2	0.0	2.55	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	2.55	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	3.55	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	5.15	600	0	600	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	1.7	6	2	1e2	0.0	1.75	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	2.9	6	2	1e2	0.0	1.75	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	1.75	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	1.7	6	2	1e2	0.0	2.55	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	2.9	6	2	1e2	0.0	2.55	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	2.55	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	3.55	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0
0.01	1	0.0	9.0	6	2	1e2	0.0	5.15	600	600	0	0	160	160	3	2	0	2000.0	1e-1	0	0	10.0	0.0

Table 6.1: Example of a complex hierachical multiparameter FWI workflow definition (modified after Kurzmann [2012]). The colors represent different parameter groups.

Chapter 7

Example 1 - the elastic Marmousi2 model

Developed in the 1990s by the French Petroleum Institute (IFP) (Versteeg [1994]) the Marmousi model is a widely used test problem for seismic imaging techniques. Beside the original acoustic version of the model an elastic version was developed by Martin et al. [2006]. This model contains both simple (approximately 1D) and complex geological structures. In the following the performance of the FWT code will be tested for the complex part of a modified Marmousi-II model using a parametrization with seismic velocities.

7.1 The complex Marmousi2 model

The Marmousi2 model (Fig. 7.1) consists of a 500 m thick water layer above an elastic subseafloor model. The sediment model is very simple near the left and right boundaries but rather complex in the centre. At both sides, the subseafloor is approximately horizontally layered, while steep thrust faults are disturbing the layers in the centre of the model. Embedded in the thrust fault system and layers are small hydrocarbon reservoirs (figure 7.1, Martin et al. [2006]).

- One shallow gas sand in a simple structural area (A).
- One relatively shallow oil sand in a structural simple area (B).
- Four faulted trap gas sands at varying depths (C1,C2,C3,C4).
- Two faulted trap oil sands at medium to deep depths (D1,D2).
- One deep oil and gas sand anticlinal trap (E1,E2).
- Water wet sand.

The deeper parts of the model consist of salt and reef structures. The thrust fault system and the reef structures are not easy to resolve by conventional first arrival tomography, so it is an ideal test model for the FWT. Due to computational restrictions the original Marmousi-II model could not be used, because the very low S-wave velocities in the sediments would require a too small spatial sampling of the model. Therefore new S-wave velocities are calculated so that the Poisson ratio is not larger than 0.25, so the soft-seabed is replaced by a hard-seabed. Sears et al. [2008] and Brossier et al. [2009] have shown the difficulties associated with soft-seabed environments for elastic FWT. Additionally the size of the Marmousi-II model is reduced from $17 \text{ km} \times 3.5 \text{ km}$ to $10 \text{ km} \times 3.48 \text{ km}$. (figure 7.2).

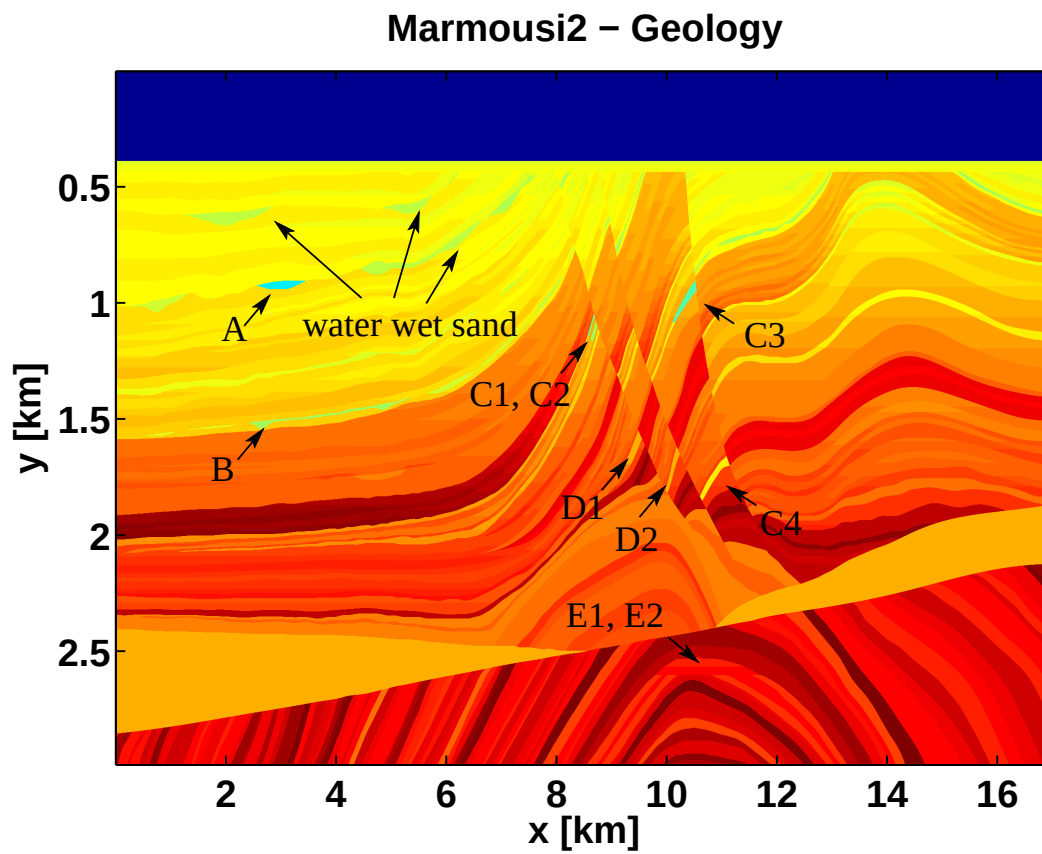


Figure 7.1: Marmousi2 model - geology.

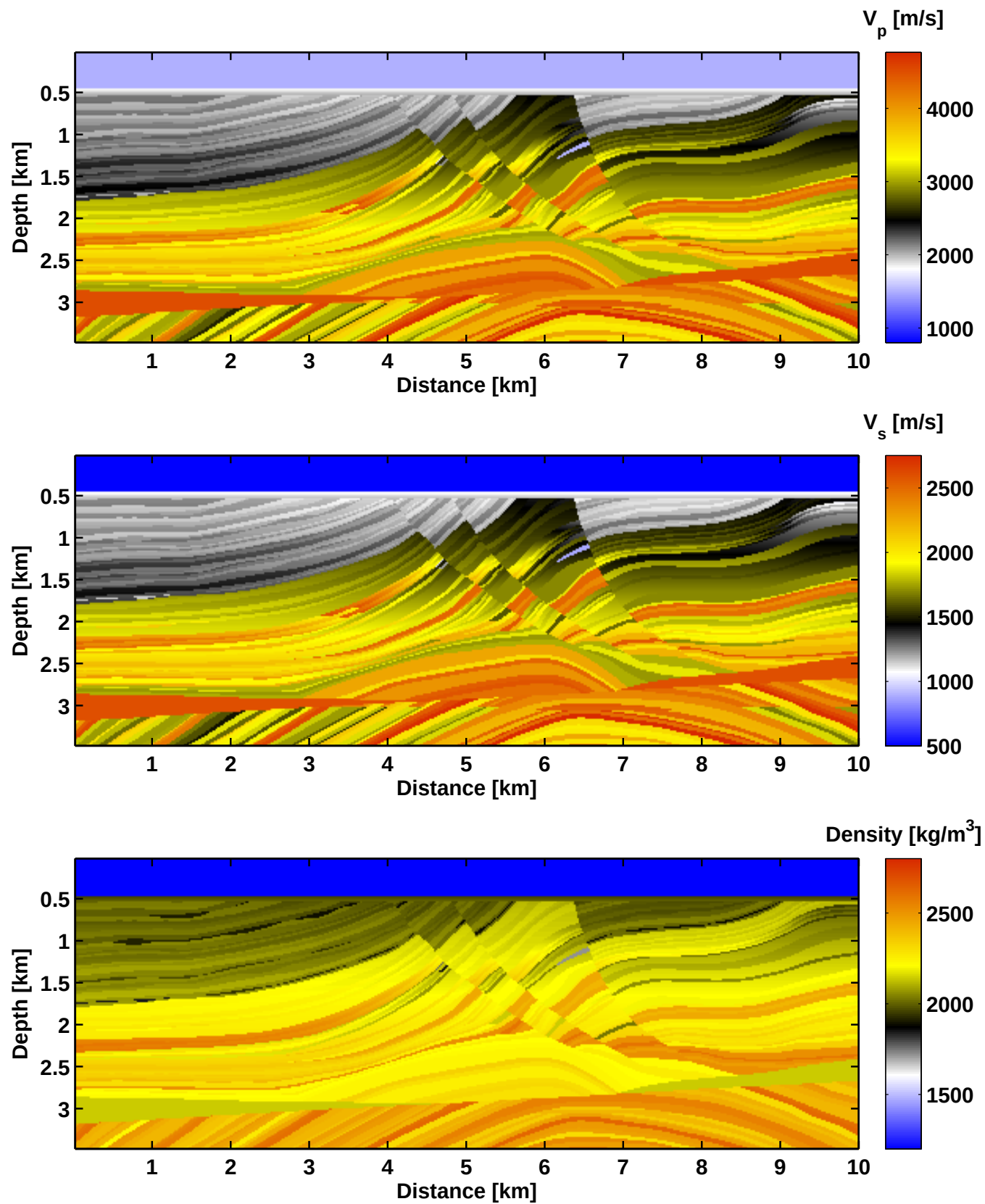


Figure 7.2: The reduced and modified complex Marmousi2 model used for the elastic FWT.

7.1.1 Acquisition geometry and FD model

The acquisition geometry consists of a Ocean-Bottom-Cable (OBC) located on the seafloor, at depth of 460 m below the free surface. The OBC contains 400 two component geophones with a spatial spacing of 20 m recording the particle velocities v_i . For the synthetic dataset 100 airgun shots are excited. The sources are towed 40 m beneath the free surface. The source signature is a butterworth low-pass filtered spike with a corner frequency of 15 Hz and order 5. The model has the dimensions $10 \text{ km} \times 3.48 \text{ km}$. Using an 8th order spatial FD operator the model can be discretized with 500×174 gridpoints in x- and z-direction with a spatial gridpoint distance of 20.0 m. The time is discretized using $\Delta T = 2.7 \text{ ms}$, thus for a recording time of $T = 6.0 \text{ s}$ 2222 time steps are needed.

7.1.2 Elastic wave propagation in the complex Marmousi model

To generate the "measured" field data for the Marmousi model copy the files listed in table 7.1 from their origin in the */benchmark* directory to their destination to define the input and elastic model parameters, as well as the acquisition geometry.

filename	origin directory	destination directory
<i>DENISE_marm_OBC.inp</i>	<i>/benchmark/Marmousi/classical_FWT/PCG/</i>	<i>/DENISE/trunk/par</i>
<i>FWI_workflow_marmousi.inp</i>	<i>/benchmark/Marmousi/classical_FWT/PCG/</i>	<i>/DENISE/trunk/par</i>
<i>receiver_OBC.dat</i>	<i>/benchmark/Marmousi/classical_FWT/receiver</i>	<i>/DENISE/trunk/par/receiver</i>
<i>source_OBC_VSP.dat</i>	<i>/benchmark/Marmousi/classical_FWT/source</i>	<i>/DENISE/trunk/par/source</i>
<i>marmousi_II_marine.vp</i>	<i>/benchmark/Marmousi/classical_FWT/start/Vel</i>	<i>/DENISE/trunk/par/start</i>
<i>marmousi_II_marine.vs</i>	<i>/benchmark/Marmousi/classical_FWT/start/Vel</i>	<i>/DENISE/trunk/par/start</i>
<i>marmousi_II_marine.rho</i>	<i>/benchmark/Marmousi/classical_FWT/start/Vel</i>	<i>/DENISE/trunk/par/start</i>
<i>marmousi_II_smooth2.vp</i>	<i>/benchmark/Marmousi/classical_FWT/start/Vel</i>	<i>/DENISE/trunk/par/start</i>
<i>marmousi_II_smooth2.vs</i>	<i>/benchmark/Marmousi/classical_FWT/start/Vel</i>	<i>/DENISE/trunk/par/start</i>
<i>marmousi_II_smooth2.rho</i>	<i>/benchmark/Marmousi/classical_FWT/start/Vel</i>	<i>/DENISE/trunk/par/start</i>

Table 7.1: Origin and destination directories of the files required for the modeling and inversion of the Marmousi model.

The optimum FD modeling (section 7.1.1) and inversion parameters are already defined in the input file *DENISE_marm_spike.inp*. However before running the forward code, check if the following parameters are set correctly:

```
READMOD=1
MFILE = start/marmousi_II_marine
INVMAT=10
```

Go to */par* directory and run the forward code on 15 cores of the cluster by typing

```
mpirun -np 15 ../bin/denise DENISE_marm_OBC.inp FWI_workflow_marmousi.inp
```

or submit a job file on the NEC-cluster. The resulting seismograms for the v_x and v_y component are written to *DENISE/trunk/par/su*. You can check the results, f.e. for the 1st shot, with the SU command

```
sugain qbal=1 < su/DENISE_MARMOUSI_y.su.shot1.it1 | suximage &
```

Generate a sub-directory for the data in *DENISE/trunk/par/su*

```
mkdir MARMOUSI_spike
```

To move the seismograms from *DENISE/trunk/par/su* to *DENISE/trunk/par/su/MARMOUSI_spike* modify the shell script *move_x_y.sh* in the */par*-directory:

```
#!/bin/csh

set x=1
while ( $x < 101)

mv su/DENISE_MARMOUSI_x.su.shot$x.it1 su/MARMOUSI_spike/DENISE_MARMOUSI_x.su.shot$x
mv su/DENISE_MARMOUSI_y.su.shot$x.it1 su/MARMOUSI_spike/DENISE_MARMOUSI_y.su.shot$x
# mv su/DENISE_MARMOUSI_p.su.shot$x.it1 su/MARMOUSI_spike/DENISE_MARMOUSI_p.su.shot$x

set x = `expr $x + 1`

end
```

Fig. 7.3 shows the development of the pressure wavefield excited by shot 50 for the central part of the complex elastic Marmousi2 model at 6 different time steps. The P wave is traveling from the source through the water column ($T=100.0$ ms) and is reflected at the seafloor ($T=400.0$ ms). In the elastic subseafloor medium the wavefield becomes very complex. The layers in the steep thrust fault system produce numerous reflections and internal multiples ($T=600.0$ ms). Additionally strong diffracted waves are generated at the sharp corners of the thrust faults between the disturbed high velocity sediment blocks within the thrust faults and the surrounding low velocity sediments. At the free surface strong multiple reflections occur ($T=800.0$ ms). The wavefront of the direct wave is quite deformed due to strong velocity contrasts within the thrust fault system. After 1500 ms nearly all kinds of waves which can be found in the literature are present: Reflections, refractions, diffractions, (internal) multiples or interface waves. The trapped gas sand reservoirs C1, C2 and C3 produce strong reflections and mode conversions. This complexity is also visible in the seismic section, recorded by the streamer in the water column. As an example Fig. 7.7, (f) shows the seismic section of the y-component for shot 50. Beside the direct wave and a strong reflection from the seafloor numerous small reflection events from the thrust fault system are dominating the seismic section.

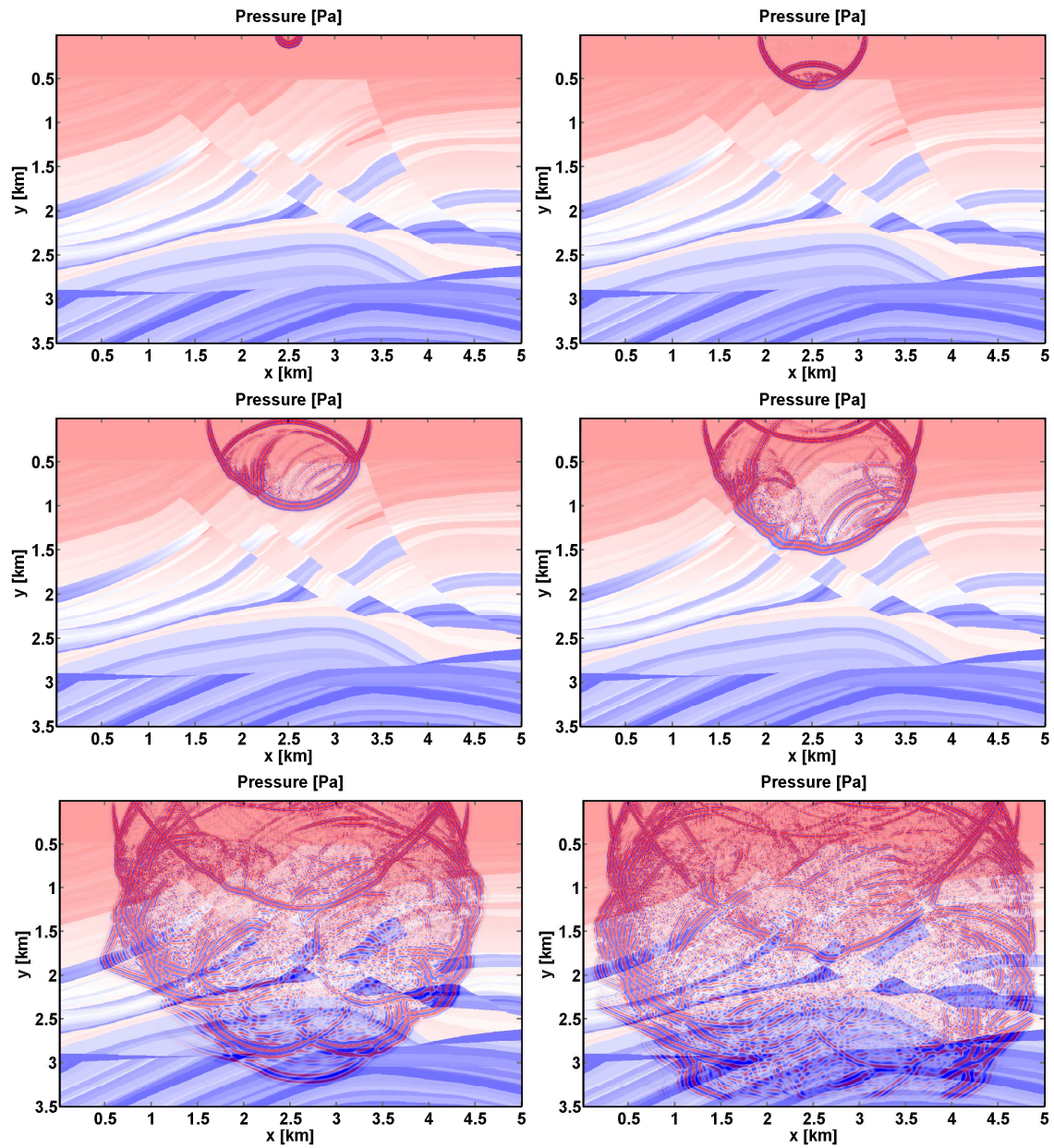


Figure 7.3: Pressure wavefield excited by shot 50 for the elastic Marmousi2 model at 6 different time steps .

7.1.3 FWI of the complex Marmousi model

Due to the far-offset acquisition geometry, we use seismic velocities as model parameters for the inversion [Köhn et al., 2012]. To generate a starting model which accurately describes the long wavelength part of the material parameters, the true models $\mathbf{m2} = [V_p, V_s, \rho]$ and $\mathbf{m1} = [\lambda, \mu, \rho]$ are filtered using a spatial 2D-Gaussian filter

$$\mathbf{m}_{\text{smooth}}(x, y) = \frac{1}{2\pi\lambda_c^2} \int_{-\lambda_c}^{\lambda_c} \int_{-\lambda_c}^{\lambda_c} dx' dy' \mathbf{m}(x - x', y - y') \exp\left(-\frac{(x - x')^2 + (y - y')^2}{2\lambda_c^2}\right) \quad (7.1)$$

with a correlation length $\lambda_c = 800.0$ m. As a result all the small scale structures vanish and only the large scale structures are present (Fig. 7.4). This starting model is comparable with the resolution of a Laplace-domain waveform inversion [Shin and Ha, 2008]. Because the Marmousi-II model is quite complicated, an additional constraint is applied during the inversion. To stabilize the inversion possible density values are restricted between 1000 kg/m^3 and 3000 kg/m^3 using hard constraints and the density step length calculated during the model update systematically reduced by a factor $\text{SCALERHO}=0.5$. Otherwise geophysically unrealistic density values might occur in the model. To increase the convergence of FWI, the inverse Hessian is approximated by a simple linear scaling of the gradient with depth. For the Marmousi-II model the inversion is separated into four parts which cover different frequency ranges, with maximum frequencies of 2, 5, 10 and 20 Hz, respectively. This inversion strategy is assembled in the workflow file *FWI_workflow_marmousi.inp*. Before running the FWI change the following parameters in the file *DENISE_marm_OBC.inp*

```
MFILE = start/marmousi_II_smooth2
INVMAT = 0
```

which defines the initial model and switches the DENISE code from modeling to inversion mode. Run the inverse code on 15 cores of the cluster by typing

```
mpirun -np 15 ../bin/denise DENISE_marm_OBC.inp FWI_workflow_marmousi.inp
```

or submit a job file on the NEC-cluster. The whole inversion requires approximately 1 day. The final inversion results after 285 iterations for the velocity parametrization are shown in Fig. 7.5. Additionally, depth profiles at $x_{p1} = 3.5$ km and $x_{p2} = 6.4$ km of the starting model and inversion result are compared with the true model in Fig. 7.6. The results contain a lot of small details. All fine layers which are completely absent in the initial model are resolved. The thrust faults and reef structures in the deeper part of the model are imaged also very well. All hydrocarbon reservoirs can be identified and even structures at the scale of the FD grid are resolved. The shear wave velocity model could also be reconstructed, even though only streamer data and therefore only P-wave information is used. This can be explained by the replacement of the soft-seabed in the original Marmousi-II model by the hard-seabed used in this study, which leads to significant P-S conversions and S-P conversions at the seafloor. Therefore the hard-seabed produces a significant footprint in the shear wave velocity model. Even the density, a parameter which can be hardly estimated from seismic data, could be recovered from the seismic wavefield. Keep in mind though, that the density image is based not only on the density information, but contains also structural V_p and V_s information due to the ambiguity investigated by the CTS test problem [Köhn et al., 2012]. The quality of the inversion results is also evident in the seismic sections of shot 50 (vertical component) plotted in Fig. 7.7. Notice the direct wave, the reflection from the ocean bottom, a few multiples and the dominant reflection from the interface between low-velocity and high-velocity sediments, but the lack of other events beyond the first arrivals in the seismic section of the starting model. The initial data residuals show no direct wave anymore, so it is fitted perfectly. The residuals only contain events from small scale model features. The seismic sections of the FWT result and the true model are nearly identical, so the final data residuals are very small. All reflection events are fitted perfectly. The normalized misfit function for the different frequency bands decreases very fast. The influence of different Hessian approximations on the inversion results is shown for the V_s model in Fig. 7.8. A PCG optimization with a simple linear scaling of the gradient with depth (top) shows a bad model resolution in the deeper parts of the model. The Hessian approximation by Plessix and Mulder [2004] with PCG (centre) improves the depth resolution. The combination with the LBFGS method (bottom) adds an additional depth resolution and overall improvement of the model sharpness.

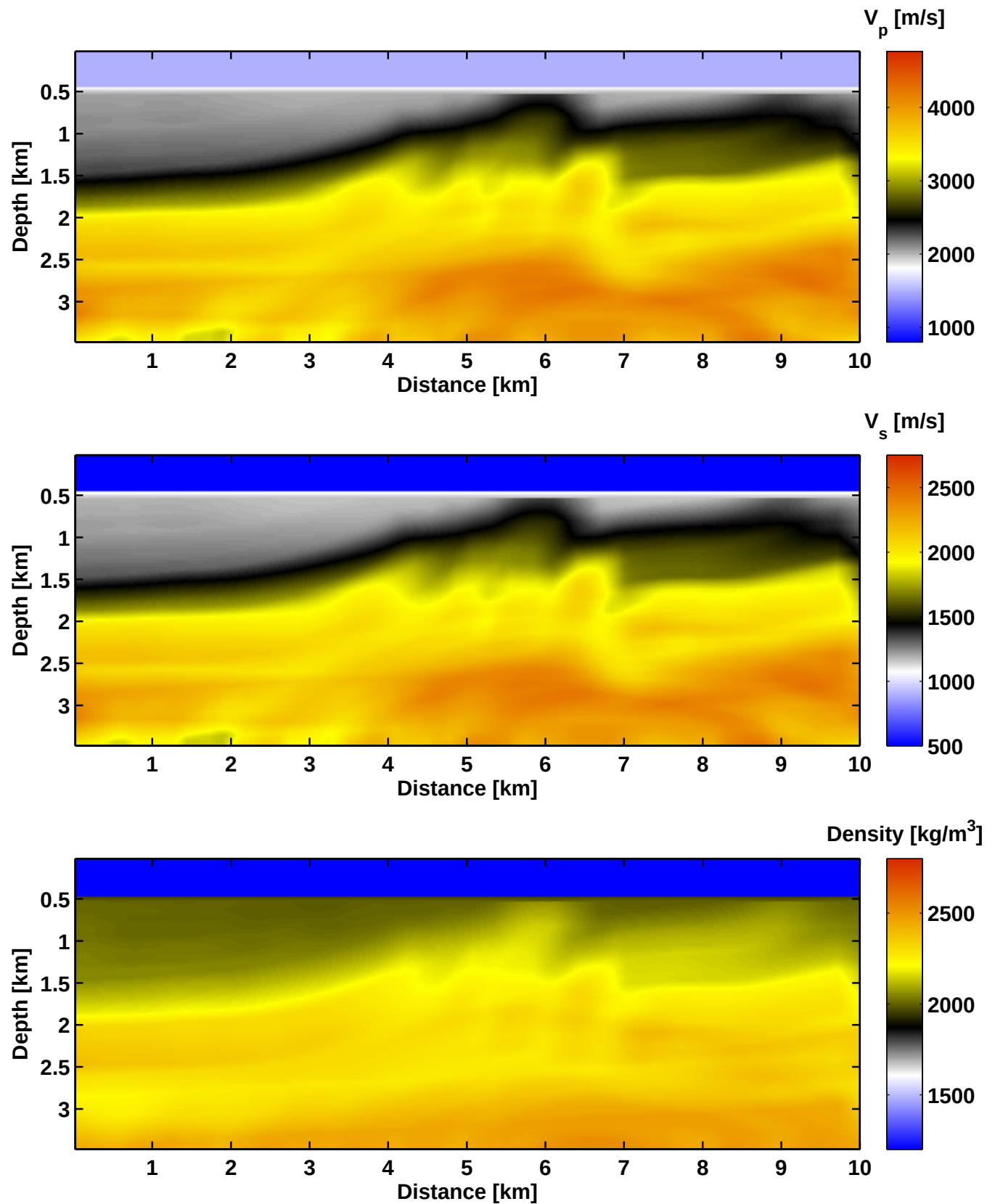


Figure 7.4: Initial models for the Marmousi-II model.

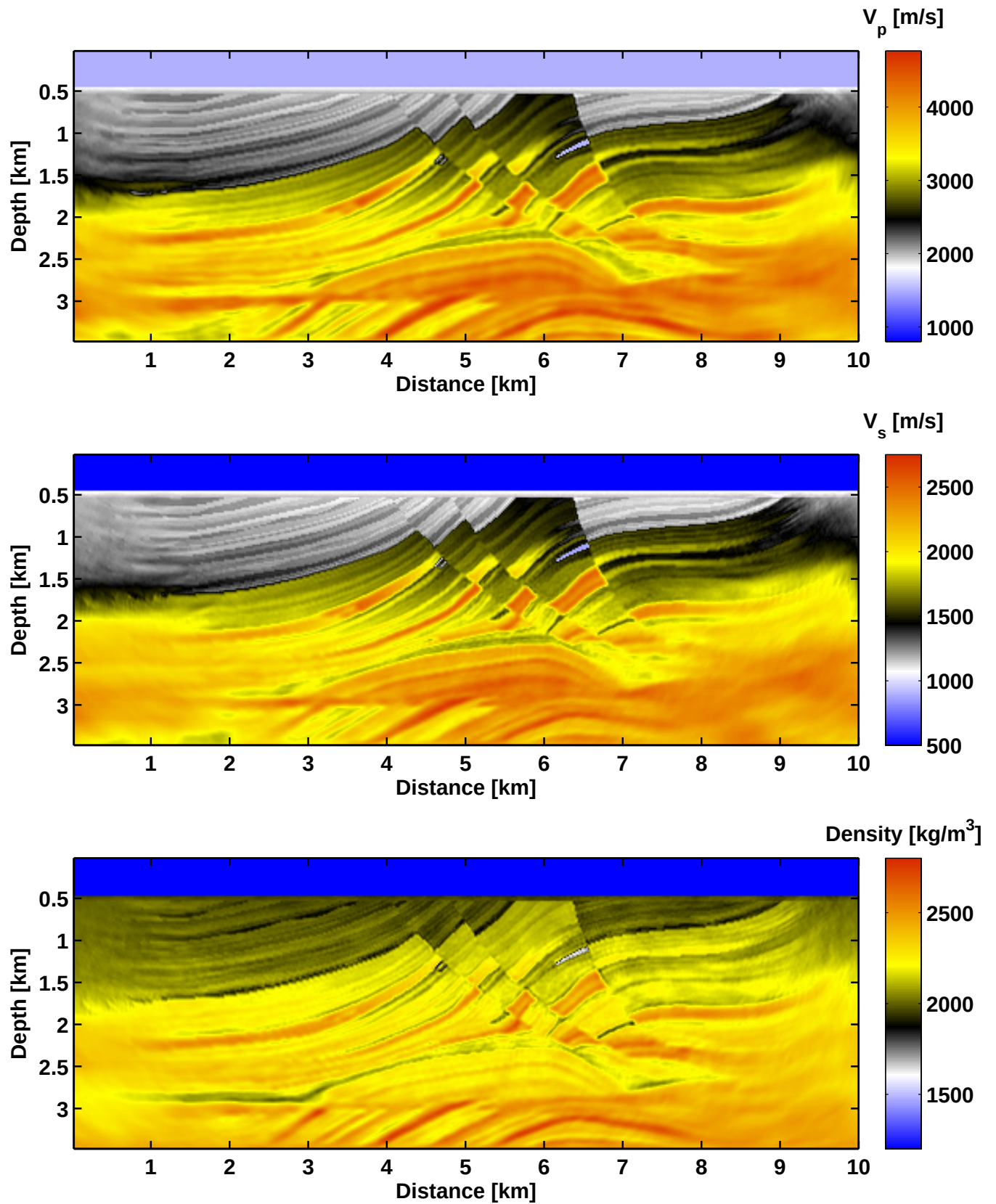


Figure 7.5: Results of the elastic FWT for the Marmousi-II model.

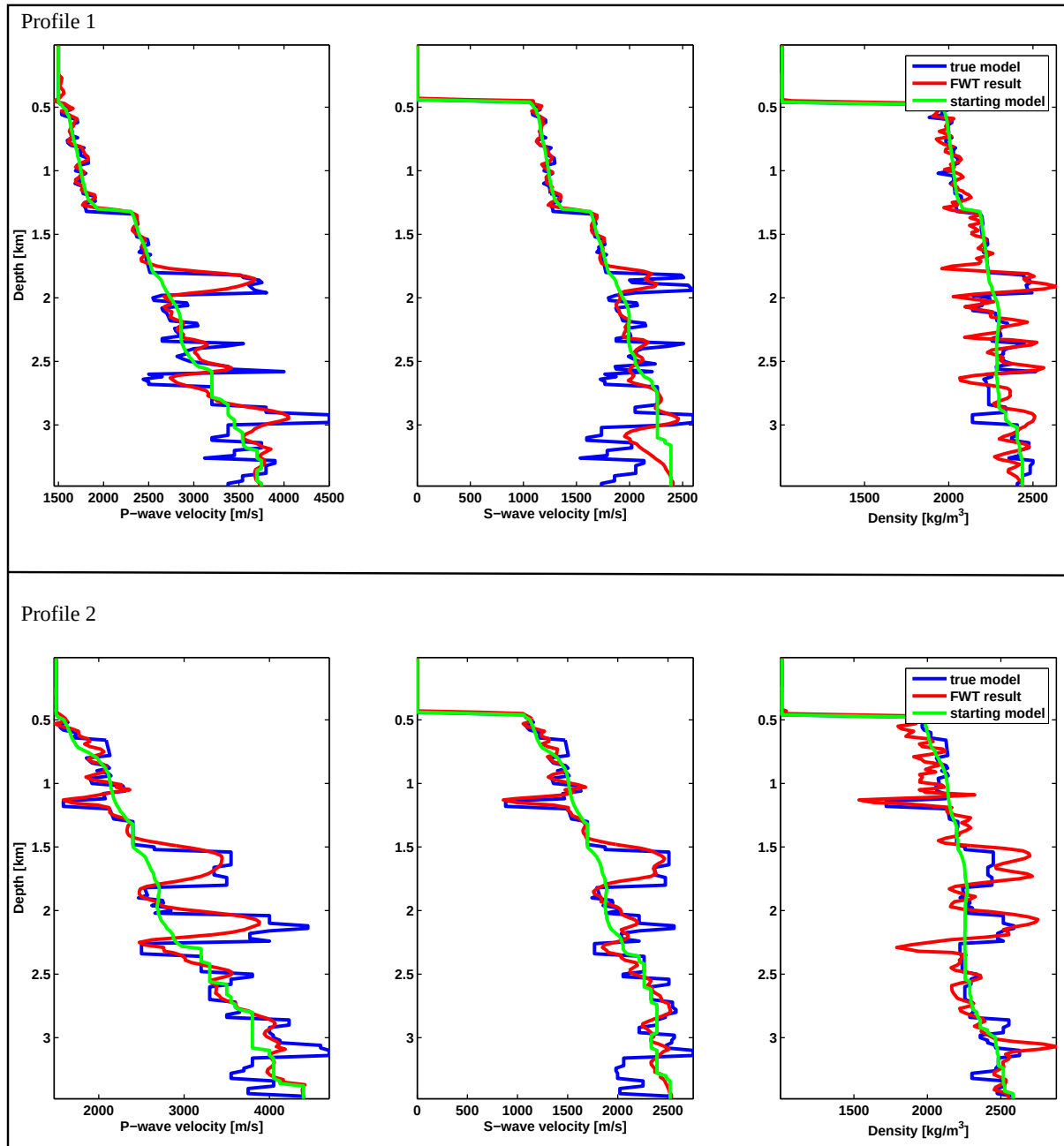


Figure 7.6: Depth profiles at $x_{p1} = 3.5$ km (top) and $x_{p2} = 6.4$ km (bottom) of the starting model and FWT result are compared with the true model for the Marmousi-II model: P-wave velocity (left), S-wave velocity (center) and density (right).

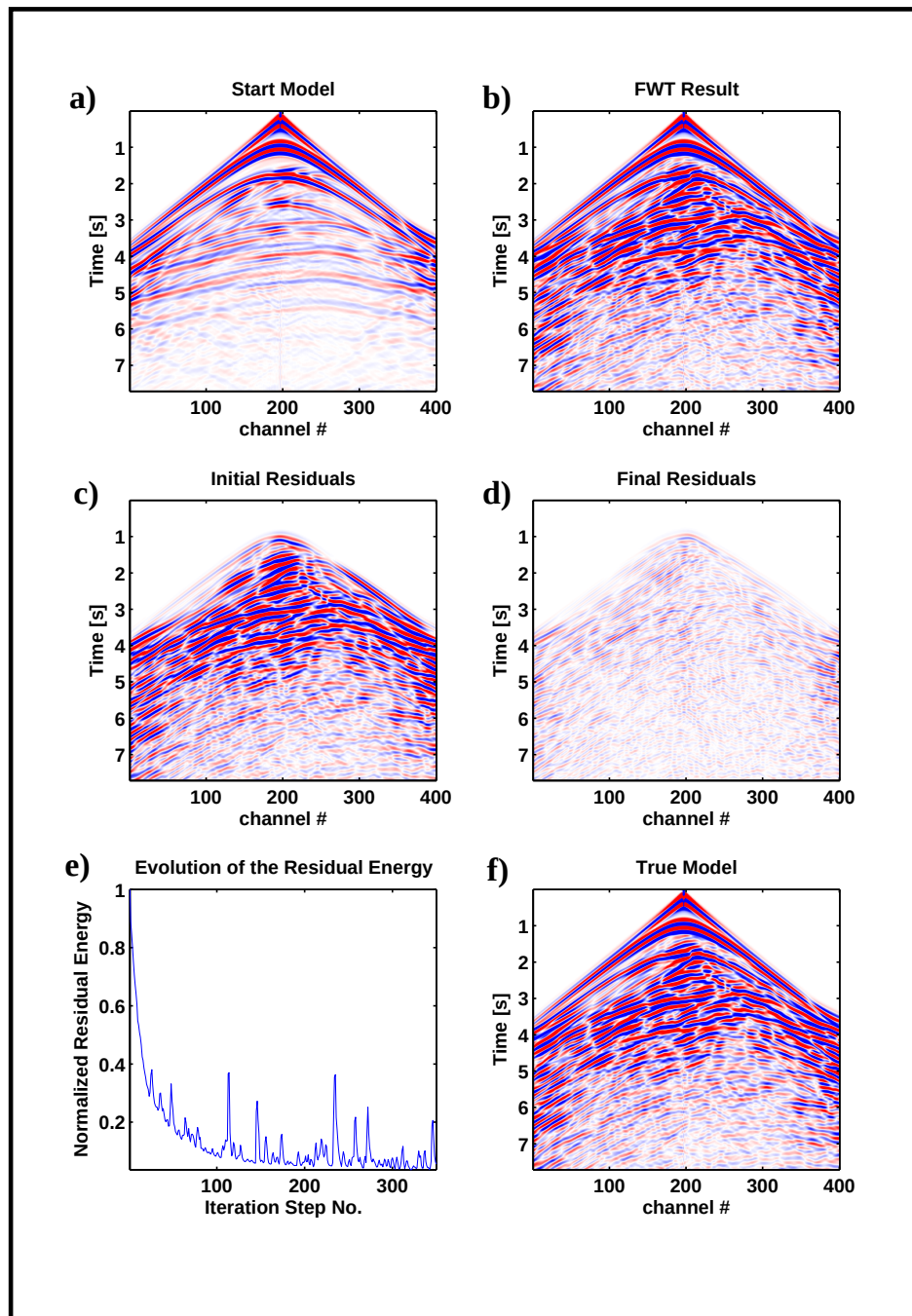


Figure 7.7: Seismic sections (shot 50, y-component) for the Marmousi-II model. (a) starting model, (b) FWT result, (c) initial residuals, (d) final residuals, (f) true model and (e) evolution of the residual energy.

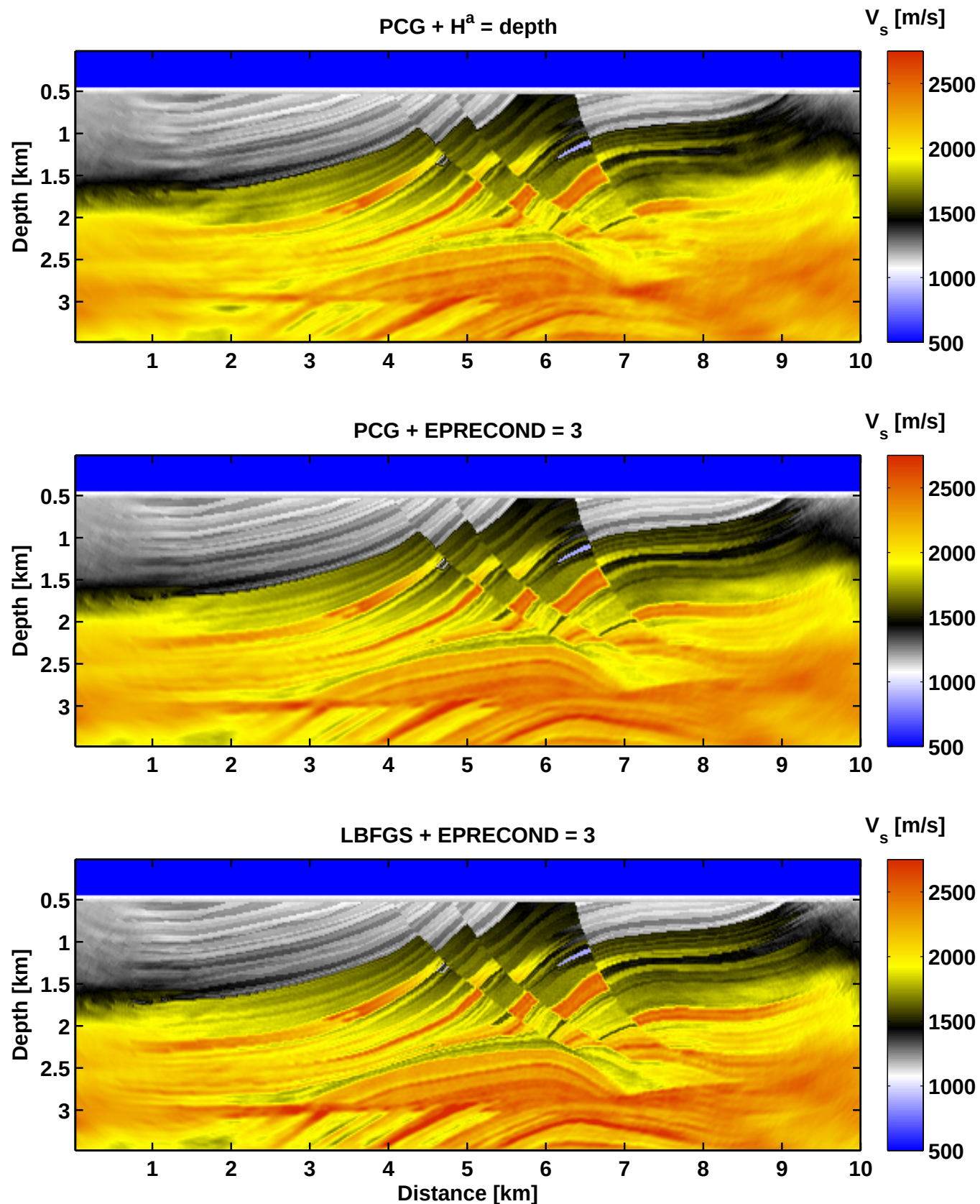


Figure 7.8: Influence of different Hessian approximations on the V_s -inversion results. A PCG optimization with a simple linear scaling of the gradient with depth (top), Hessian approximation by Plessix and Mulder [2004] with PCG (centre) and LBFGS (bottom).

7.1.4 Marmousi-II Benchmarks

Due to the massive amount of required forward models the Marmousi-II inversion is a perfect benchmark problem. A few representative benchmark results are shown in table 7.3. Due to continuous bugfixing and improvements in the source code, the computation times are not comparable.

RZ Kiel Linux-Cluster	Intel Harpertown nodes (2.33 GHz)	8 cores/node	8 GB RAM/node	Gigabit Ethernet	
Compiler	MPI	nodes	cores	cores - total	computation time
PGI-11.4	MPICH2	4	8	32	3.5 d
	Intel Westmere nodes (2.67 GHz)	12 cores/node	48 GB RAM/node	Gigabit Ethernet	
Compiler	MPI	nodes	cores	cores - total	computation time
Intel 14.0.0	Intel-MPI 4.1	1	8	8	20.3 h

Table 7.3: Marmousi-II benchmark results.

Bibliography

- K. Aki and P. Richards. *Quantitative seismology*. W.H. Freeman and Company, 1980.
- S. al Hagrey, D. Köhn, and W. Rabbel. Geophysical assessments of renewable gas energy compressed in geologic pore storage reservoirs. *SpringerPlus*, 3(1):1–16, 2014.
- J. Blanch, J. Robertsson, and W. Symes. Modeling of a constant Q: Methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique. *Geophysics*, 60(1):176–184, 1995.
- T. Bohlen. *Interpretation of Measured Seismograms by Means of Viscoelastic Finite Difference Modelling*. PhD thesis, Kiel University, 1998.
- T. Bohlen. Parallel 3-D viscoelastic finite-difference seismic modelling. *Computers & Geosciences*, 28(8):887–899, 2002.
- T. Bohlen and E. Saenger. Accuracy of heterogeneous staggered-grid finite-difference modeling of Rayleigh waves. *Geophysics*, 71(4):T109–T115, 2006.
- A. Brenders and R. Pratt. Full waveform tomography for lithospheric imaging: results from a blind test in a realistic crustal model. *Geophys. J. Int.*, 168:133–151, 2007.
- R. Brossier. *Imagerie sismique à deux dimensions des milieux visco-élastiques par inversion des formes d’ondes : développements méthodologiques et applications*. PhD thesis, Université de Nice - Sophia Antipolis, 2009.
- R. Brossier, S. Operto, and J. Virieux. *Two-dimensional seismic imaging of the Valhall model from synthetic OBC data by frequency-domain elastic full-waveform inversion*, chapter 461, pages 2293–2297. 2009.
- Y. Choi and T. Alkhalifah. Application of multi-source waveform inversion to marine streamer data using the global correlation norm. *Geophysical Prospecting*, 60:748–758, 2012.
- Y. Choi, D. Min, and C. Shin. Frequency-domain elastic full waveform inversion using the new pseudo-hessian matrix: Experience of elastic marmousi-2 synthetic data. *Bull., Seis Soc. Am.*, 98:2402–2415, 2008a.
- Y. Choi, D. Min, and C. Shin. Two-dimensional waveform inversion of multi-component data in acoustic-elastic coupled media. *Geophysical Prospecting*, 56:863–881, 2008b.
- R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100:32–74, 1928.
- R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM Journal*, pages 215–234, March 1967.
- H. Denli and L. Huang. Double-difference elastic waveform tomography in the time domain. In *SEG Technical Program Expanded Abstracts*, pages 2302–2306, 2009.
- M. Dougherty and R. Stephen. Seismic energy partitioning and scattering in laterally heterogeneous ocean crust. *Pure Appl. Geophys.*, 128(1/2):195 – 239, 1988.
- R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, 1964.

- O. Gauthier, J. Virieux, and A. Tarantola. Two-dimensional nonlinear inversion of seismic waveforms - numerical results. *Geophysics*, 51(7):1387–1403, 1986.
- M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.
- O. Holberg. Computational aspects of the choice of operator and sampling interval for numerical differentiation in large-scale simulation of wave phenomena. *Geophysical Prospecting*, 35:629–655, 1987.
- C. Jastram. *Seismische Modellierung mit Finiten Differenzen höherer Ordnung auf einem Gitter mit vertikal variierendem Gitterabstand*. PhD thesis, Universität Hamburg, 1992.
- D. Köhn. *Time Domain 2D Elastic Full Waveform Tomography*. PhD thesis, Kiel University, 2011. available at <http://nbn-resolving.de/urn:nbn:de:gbv:8-diss-67866>.
- D. Köhn, D. De Nil, A. Kurzmann, A. Przebindowska, and T. Bohlen. On the influence of model parametrization in elastic full waveform tomography. *Geophysical Journal International*, 191(1):325–345, 2012.
- D. Komatitsch and R. Martin. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, 72(5):155 – 167, 2007.
- A. Kurzmann. *Applications of 2D and 3D full waveform tomography in acoustic and viscoacoustic complex media*. PhD thesis, Karlsruhe Institute of Technology (KIT), 2012. available at <http://nbn-resolving.de/urn:nbn:de:swb:90-344211>.
- A. Levander. Fourth-order finite-difference P-SV seismograms. *Geophysics*, 53(11):1425–1436, 1988.
- G. Martin, R. Wiley, and K. Marfurt. Marmousi2 - An elastic upgrade for Marmousi. *The Leading Edge*, 25:156–166, 2006.
- P. Moczo, J. Kristek, and L. Halada. *The Finite-Difference Method for Seismologists. An Introduction*. Comenius University, Bratislava, 2004.
- P. Mora. Nonlinear two-dimensional elastic inversion of multioffset seismic data. *Geophysics*, 52:1211 – 1228, 1987.
- P. Morse and H. Feshbach. *Methods of theoretical physics*. McGraw-Hill Book Company, New York, 1953.
- J. Nocedal and S. Wright. *Numerical Optimization*. Springer, New York, 2006.
- A. Pica, J. Diet, and A. Tarantola. Nonlinear inversion of seismic reflection data in a laterally invariant medium. 55 (3):284–282, 1990.
- R.-E. Plessix and W. A. Mulder. Frequency-domain finite-difference amplitude-preserving migration. *Geophysical Journal International*, 157(3):975–987, 2004.
- E. Polak and G. Ribière. Note sur la convergence de méthodes de directions conjuguées. *Revue Francaise d'Informatique et de Recherche Opérationnelle*, 16:35–43, 1969.
- R. Pratt. Inverse theory applied to multi-source cross-hole tomography. Part II: Elastic wave-equation method. *Geophysical Prospecting*, 38:311–329, 1990.
- R. Pratt. Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale model. *Geophysics*, 64:888–901, 1999.
- R. Pratt. Velocity models from frequency-domain waveform tomography: Past, present and future. In *66th EAGE conference and exhibition, Expanded Abstracts*, pages 181–182, Paris, France, 2004.
- R. Pratt and M. Worthington. Inverse theory applied to multi-source cross-hole tomography. Part I: Acoustic wave equation method. *Geophysical Prospecting*, 38:287–310, 1990.

- R. Pratt, F. Gao, C. Zelt, and A. Levander. The limits and complementary nature of traveltime and waveform tomography. In *International Conference of Sub-basalt imaging, Expanded Abstracts*, pages 181–182, Cambridge, England, 2002.
- R. Pratt, L. Huang, N. Duric, and P. Littrup. Sound-speed and attenuation imaging of breast tissue using waveform tomography of transmission ultrasound data. 2007.
- J. Robertsson, J. Blanch, and W. Symes. Viscoelastic finite-difference modeling. *Geophysics*, 59(9):1444–1456, 1994.
- J. Robertsson, A. Levander, W. Symes, and K. Holliger. A comparative study of free-surface boundary conditions for finite-difference simulation of elastic/viscoelastic wave propagation. pages 1277–1280, Houston, Texas, 1995.
- H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3: 175–184, 1960.
- T. Sears, S. Singh, and P. Barton. Elastic full waveform inversion of multi-component OBC seismic data. *Geophysical Prospecting*, 56(6):843–862, 2008.
- D. Sheen, K. Tuncay, C. Baag, and P. Ortaleva. Time domain Gauss-Newton seismic waveform inversion in elastic media. *Geophys. J. Int.*, 167:1373–1384, 2006.
- C. Shin and W. Ha. A comparison between the behavior of objective functions for waveform inversion in the frequency and Laplace domains. *Geophysics*, 73(5):VE119–VE133, 2008.
- C. Shin, K. Yoon, K. Marfurt, K. Park, D. Yang, H. Lim, S. Chung, and S. Shin. Efficient calculation of a partial-derivative wavefield using reciprocity for seismic imaging and inversion. *Geophysics*, 6:1856–1863, 2001.
- R. Shipp and S. Singh. Two-dimensional full wavefield inversion of wide-aperture marine seismic streamer data. *Geophys. J. Int.*, 151:325–344, 2002.
- F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J. L'Excellent. FWT2D: a massively parallel program for frequency domain full-waveform tomography of wide-aperture seismic data - part 1: algorithm. *Computer & Geosciences*, 35:487–496, 2009a.
- F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J. L'Excellent. FWT2D: a massively parallel program for frequency domain full-waveform tomography of wide-aperture seismic data - part 2: numerical examples and scalability analysis. *Computer & Geosciences*, 35:496–514, 2009b.
- A. Tarantola. Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, 49:1259–1266, 1984a.
- A. Tarantola. Linearized inversion of seismic reflection data. *Geophysical Prospecting*, 32:998–1015, 1984b.
- A. Tarantola. A strategy for nonlinear elastic inversion of seismic reflection data. *Geophysics*, 51:1893–1903, 1986.
- A. Tarantola. Theoretical background for the inversion of seismic waveforms, including elasticity and attenuation. *PAGEOPH*, 128:365–399, 1988.
- A. Tarantola. *Inverse Problem Theory*. SIAM, 2005.
- R. Versteeg. The marmousi experience: Velocity model determination on a complex data set. *The Leading Edge*, 13: 927–936, 1994.
- J. Virieux. P-SV wave propagation in heterogeneous media: velocity-stress finite-difference method. *Geophysics*, 51 (4):889–901, 1986.